PROGRAMANDO CON sinclair QL

indescomp

Librería TAJO
Merced, 11-Tel. 24 44 69
MURCIA

Programando con QL

Programando con QL

Ian Sinclair

PROGRAMANDO CON QL

Edición española de la obra

QL COMPUTING
Ian Sinclair

publicada en castellano bajo licencia de

Granada Technical Books
Granada Publishing Ltd
8 Grafton Street, London W1X 3LA

Traducción:
Emilio Benito Santos

INDESCOMP, S. A.
Avda. del Mediterráneo, 9
28007 Madrid

© 1984 Ian Sinclair
© 1985 Indescomp, S. A.

Reservados todos los derechos. Prohibida la reproducción total o parcial de la obra, por cualquier medio, sin el permiso escrito de los editores.

ISBN 84 86176 **22** 0
Depósito Legal: M-2536-85

Impresión: Gráficas Lormo. Isabel Méndez, 15. Madrid.

Producción de la edición española:
Vector Ediciones
Gutierre de Cetina, 61
28017 Madrid

Contenido

Prólogo VII
1 Instalación del QL 1
2 Manejo de la pantalla 19
3 Festival de variables 33
4 Repetición de tareas 49
5 Cadenas literales 67
6 Menúes, subrutinas y programas 85
7 Canales, microdrives y archivo de datos 103
8 Ventanas y otros efectos 127
9 Introducción a los gráficos 141
10 Los sonidos del QL 155
Apéndice A: Edición 167
Apéndice B: Conexión de una impresora 169
Índice 171

Prólogo

La memoria del QL es mucho más amplia que la de las demás máquinas de su nivel de precio; por ello es de esperar que llegue a ocupar un lugar importante entre los ordenadores de uso doméstico. Aunque el fabricante afirme que los programas que se entregan con la máquina han sido diseñados para su utilización en las empresas, no parece que el sistema de microdrive pueda atraer al usuario profesional tanto como al aficionado. Estos programas, por otra parte, sólo se pueden aprovechar plenamente si el usuario ha instalado el ordenador correctamente y comprende su funcionamiento, para lo cual este libro puede serle muy útil.

Además, es indudable que habrá aplicaciones del QL que no quedarán cubiertas con los programas suministrados. Cuando los conocimientos del lector hayan alcanzado la fase en la que sea capaz de diseñar los programas adecuados para resolver sus propios problemas, necesitará saber cómo programar el QL. La programación es, después de todo, la principal tarea para la que se compra un ordenador. Tener un ordenador y no programarlo es como comprar un Porsche y hacer que lo conduzca un chófer. Si el lector no ha programado nunca un ordenador, este libro le enseñará a hacerlo. En cambio, si ha conocido otros ordenadores de la misma marca, como por ejemplo el ZX, este libro le dará la clave de muchas posibilidades que son exclusivas del QL. El BASIC del QL es muy diferente del de otras máquinas, tanto que es necesario aprenderlo partiendo de cero.

Debo resaltar que he escrito este libro *utilizando* un QL, y que los programas que ofrezco como ejemplos han sido comprobados en él. Esta afirmación puede parecer innecesaria, pero es un hecho que se han escrito libros sobre el QL *antes* de que este ordenador estuviera en el mercado. Todas las afirmaciones de este libro han sido comprobadas; no he copiado nada del manual sin la adecuada comprobación. En los casos en los que una orden o una instrucción no funcionan como el manual asegura, he hecho notar la diferencia. Los primeros modelos del QL tienen parte de sus circuitos en un dispositivo externo que se conecta al ordenador. Es de esperar que haya dife-

rencias entre estas primeras versiones y la versión final, pero no creo que tales diferencias afecten sustancialmente al contenido del libro; en la versión final seguramente se incluirán instrucciones nuevas, pero no creo que tales diferencias afecten sustancialmente al contenido del libro; en la versión final seguramente se incluirán instrucciones nuevas, pero no se suprimirá ninguna de las actuales. El libro lo he desarrollado utilizando un televisor, pues no creo que sean muchos los usuarios que se compren un monitor.

Como siempre, estoy muy agradecido a las muchas personas que han hecho posible este libro. La máquina me fue enviada por Sinclair Reasearch; Richard Miles, de Granada Technical Books, trabajó incansablemente para conseguir que el ordenador estuviera a mi disposición lo más pronto posible. Sue Moore, además de otras miembros del personal de Granada Technical Books, ha hecho maravillas con el manuscrito. Los teclistas de fotocomposición y los impresores, los más eficaces que conozco, han conseguido que el libro viera la luz en un tiempo récord. Espero que toda esta labor de equipo haya dado como resultado un libro que esté a la altura de la calidad del ordenador al que va dedicado.

Conviene aclarar que, aparte la coincidencia del nombre, no tengo ninguna relación con Sinclair Reasearch ni con Sir Clive Sinclair.

Ian Sinclair

1

Instalación del QL

El QL viene excepcionalmente bien embalado. La caja contiene el ordenador en sí, la fuente de alimentación, el cable para TV, el cable de red, un cable para impresora serie (incluido en las primeras versiones) y un voluminoso manual. A menos que el lector tenga cierta experiencia en el manejo de ordenadores, necesitará más información que la que el manual puede ofrecerle, a pesar de que este manual es bastante mejor que el de la mayor parte de los ordenadores. La fuente de alimentación es el aparato negro de forma cúbica que tiene dos juegos de cables. Uno de los cables se conecta permanentemente a un zócalo de tres terminales que está situado en la cara posterior de la carcasa del teclado, a la derecha. Este zócalo está marcado con la palabra «POWER». La clavija debe ser insertada en la posición correcta, con la parte biselada hacia arriba. No la fuerce; el encaje debe ser suave, aunque firme. Si fuerza la clavija, conseguirá insertarla boca abajo, y esto puede dañar el ordenador. La fuente de alimentación es muy pesada; si levanta el teclado, levante también la fuente de alimentación o, preferiblemente, desconéctela. En el extremo del otro cable hay que instalar la clavija adecuada antes de poder conectar el aparato a la red.

La clavija se conecta internamente según se indica en la figura 1.1. Sólo hay dos hilos, uno azul y otro marrón. El cable debe quedar firmemente sujeto con la abrazadera. El fusible debe ser de 3 amperios, no de los de 13 amperios que se suelen instalar en las clavijas de tres terminales. Si está usted habituado a preparar sus propios enchufes, el diagrama le será suficiente indicación. En caso contrario, haga que se lo conecte un electricista (con el fusible de 3 amperios). El electricista no necesita el ordenador completo, sino solamente la fuente de alimentación. Una vez preparado el enchufe, asegúrese de que la fuente de alimentación esté alejada del QL y del televisor. Póngala de forma que el aire pueda circular libremente a su alrededor. Sólo se calienta moderadamente cuando está conectado el ordenador, pero en cual-

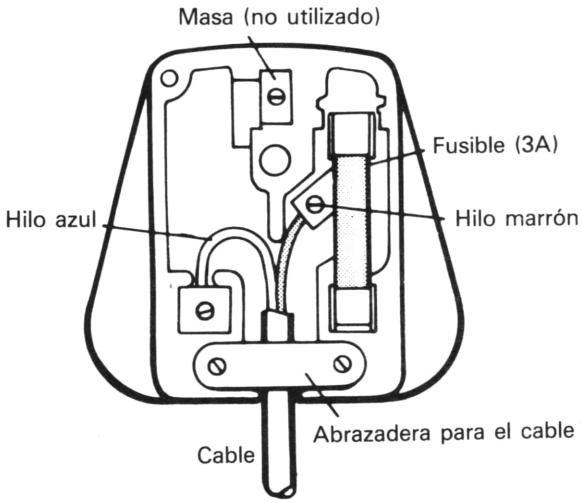

Fig. 1.1 Conexiones en la clavija de red de tres terminales. Sólo se conectan la fase y el neutro, no la masa. Si no tiene experiencia en este tipo de trabajo, es preferible que lo confíe a un electricista.

quier caso conviene facilitarle la refrigeración. Le sugerimos que la coloque sobre un trozo de fieltro o de poliestireno; cuando se la deja sobre una superficie rígida produce un zumbido muy molesto.

Si ha seguido estas instrucciones, ya está casi preparado para hacer funcionar el QL; sólo necesita un televisor o un monitor. El ordenador genera señales eléctricas con las que se forman imágenes en la pantalla. Esto se puede hacer de dos formas. Una de ellas es conectar un *monitor*, que es un dispositivo especialmente diseñado para utilizar directamente las señales procedentes del ordenador; normalmente los monitores no sirven para recibir señales de TV. Otra forma es convertir las señales del ordenador en señales del mismo tipo que las emitidas por los transmisores de TV, de manera que puedan ser introducidas por la entrada de antena de un televisor doméstico.

Hay una diferencia muy importante entre estos dos tipos de señales. Aunque puede ser cómodo (y más barato) utilizar un televisor, la imagen que se obtiene es de baja calidad. Esto ocurre porque los receptores de televisión no han sido diseñados para manejar señales de ordenador y por la necesidad de convertir éstas en señales compatibles con el televisor. El resultado es que las letras y otros caracteres parecen borrosos, y los colores como formados por rayas. En cambio, si conecta un monitor, apreciará que los signos que aparecen en la pantalla son mucho más nítidos y que los colores son mejores. En esta conexión puede necesitar la ayuda del establecimiento en el que ha

adquirido su ordenador, pues el cable para monitor de Sinclair (extra) sólo tiene una clavija, la que se conecta al ordenador. El personal de la tienda podrá instalar la clavija requerida por el monitor.

VER PARA CREER

La única forma de saber qué está haciendo el ordenador en cada momento es tenerlo conectado a un monitor o a un televisor. Para el funcionamiento del ordenador esto es indiferente, pero el usuario necesita ver en la pantalla qué está ocurriendo. Si va a utilizar un televisor, tendrá que conectar el cable de antena tanto al ordenador como al televisor. A menos que pueda dedicar un televisor exclusivamente al ordenador, pronto se verá enchufando y desenchufando el cable de antena con demasiada frecuencia. Esto no es ni agradable ni conveniente, pues se corre el riesgo de deteriorar los contactos del zócalo del televisor. Una posibilidad que recomendamos es utilizar un adap-

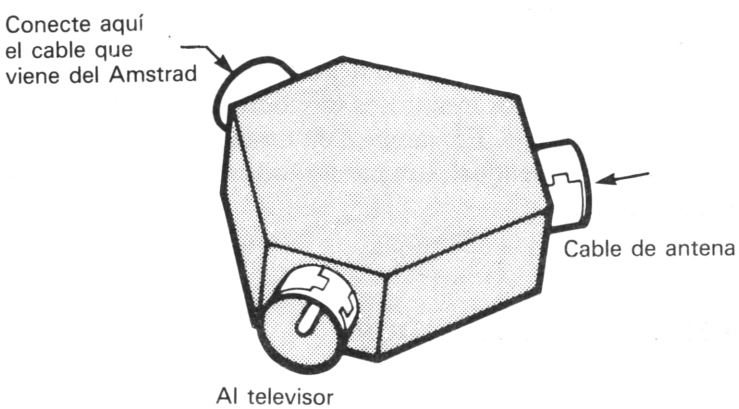

Fig. 1.2 Adaptador en «Y» para la entrada de antena del televisor; permite tener conectados simultáneamente el ordenador y la antena.

tador en «Y», como el que se ilustra en la figura 1.2. Este adaptador permite tener conectados simultáneamente el cable procedente de la antena y el del ordenador. La conexión del QL al televisor o al adaptador en «Y» se realiza mediante el cable que se suministra con el QL. Este cable tiene dos clavijas distintas (Fig. 1.3). Si ha instalado el adaptador en «Y», todo lo que tiene que hacer para cambiar del ordenador a la televisión es conmutar los canales.

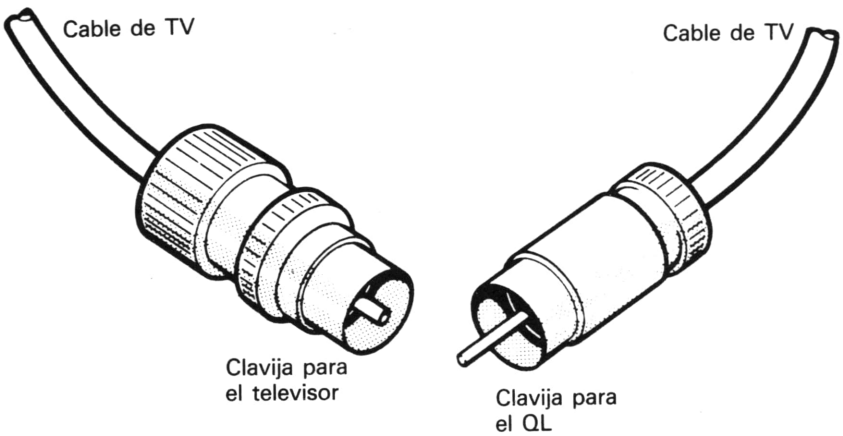

Fig. 1.3 Clavijas del cable de TV.

El televisor que dedique al QL no tiene que ser necesariamente de color, al menos para empezar. Para aprender a programar el QL no necesita ver los resultados en color mientras no lleguemos al capítulo 8. En un televisor de blanco y negro (y en los monitores monocromáticos) los colores aparecen como gradaciones de gris. En cambio, en un televisor de color podrá ver los colores en todo su esplendor, aunque no todos los televisores dan imágenes de la misma calidad.

ENCENDIDO DEL ORDENADOR

Antes de empezar a enchufar todo lo que tiene a la vista, compruebe de cuántas tomas de corriente dispone. Para el QL necesitará al menos dos, una para la fuente de alimentación y otra para el televisor. La mayor parte de las viviendas están escasas de enchufes; puede ser necesario, sobre todo cuando adquiera una impresora u otros periféricos, que se construya o compre una regleta de tomas de corriente como la que se muestra en la figura 1.4. Esta regleta le permite organizar mejor las conexiones, y contribuye a evitar el desastre que representaría tirar el ordenador al suelo si tropezara con un cable. No confíe en los «ladrones» o enchufes de tres vías, en los que los contactos nunca son seguros. El QL no tiene interruptor, de modo que *siempre* se debe desenchufar el cable de alimentación al terminar una sesión de trabajo. El ruido que produce la fuente de alimentación le servirá de recordatorio.

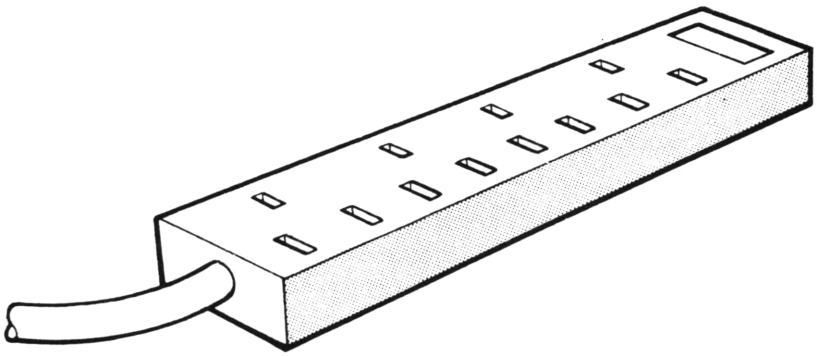

Fig. 1.4 Regleta de cuatro tomas, preferible a los «ladrones» o enchufes múltiples.

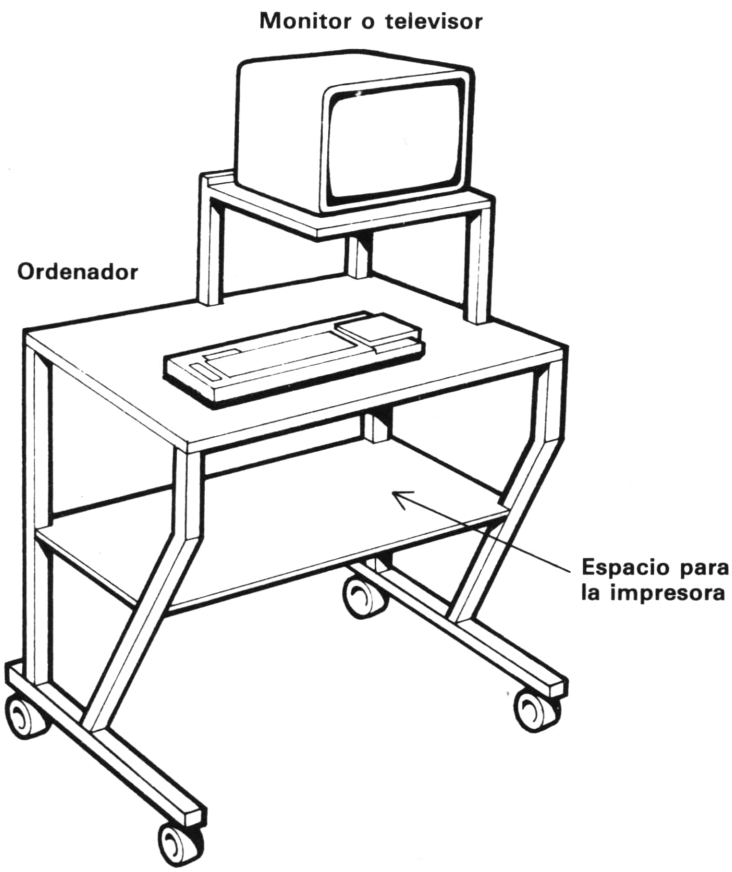

Fig. 1.5 Mueble recomendado para instalar ordenadamente el QL y sus periféricos.

Ahora que ya tiene el equipo esencial (teclado, fuente de alimentación y monitor o televisor) preparado, necesitará una superficie de trabajo bastante amplia. Más adelante puede necesitar una impresora, unidades de disquette y otros periféricos (con lo que dispondrá de un *sistema* informático, no simplemente de un ordenador). Para todos estos aparatos hace falta espacio. Le podemos recomendar que adquiera un mueble de diseño especial, como el que se ilustra en la figura 1.5. Si no quiere llegar a tanto, una mesa suficientemente grande puede servirle por ahora. La informática es como la alta fidelidad: siempre hay algo nuevo que comprar.

Ahora que ya lo tiene todo instalado y conectado, le haremos algunas advertencias. A diferencia de otros ordenadores domésticos de menor precio, que graban la información en cintas magnéticas en cassette, el QL dispone de un sistema especial de cinta denominado *microdrive* (palabra que se pronuncia «maicrodraiv»). La cinta va alojada en unos cartuchos que son caros y considerablemente frágiles. Estos cartuchos se suministran con una caja protectora. Para utilizarlos se los inserta en las ranuras dispuestas al efecto en la cara frontal del QL, a la derecha. No intente introducirlos con la máquina apagada. Una precaución muy importante es la siguiente: *nunca conecte ni desconecte la máquina teniendo el cartucho insertado*, pues es *posible* que se pierda la información grabada en él. En honor a la verdad, debo decir que, aunque este riesgo exista, a mí no me ha ocurrido nunca en las severas pruebas a que he sometido este ordenador.

Los primeros QL comercializados (y el mío es uno de ellos) venían con una pequeña caja que debía ser insertada en el zócalo marcado con la palabra «ROM». Este zócalo está en la cara posterior del teclado, al lado izquierdo, y la ROM se coloca con la etiqueta hacia arriba. Pues bien, *la ROM debe ser instalada antes de encender el ordenador*. Es inútil tratar de hacer funcionar la máquina sin la ROM, pues en ella están grabadas las instrucciones que constituyen la parte «inteligente» del ordenador. En los modelos posteriores la ROM ya viene instalada dentro de la carcasa.

Nuestra siguiente etapa va a consistir en encender el QL y el televisor. Si lo que utiliza es un televisor, lo normal es que en principio en la pantalla no se vea absolutamente nada. La razón es que el televisor tiene que ser sintonizado para recibir las señales del ordenador. Vamos a ver, pues, cómo se realiza esta sintonía.

En la figura 1.6 se muestran los tres métodos más frecuentes de sintonía de televisores. El más sencillo es el de la figura 1.6*a*. Éste es el tipo que se encuentra en los televisores portátiles de blanco y negro. Para visualizar las señales del QL basta con girar el mando de sintonía. Si éste está marcado con números, pruebe en la zona comprendida entre el 30 y el 40. Si no lo está, gire el mando completamente en un sentido, y luego gírelo lentamente en sentido contrario hasta que aparezca la imagen en la pantalla.

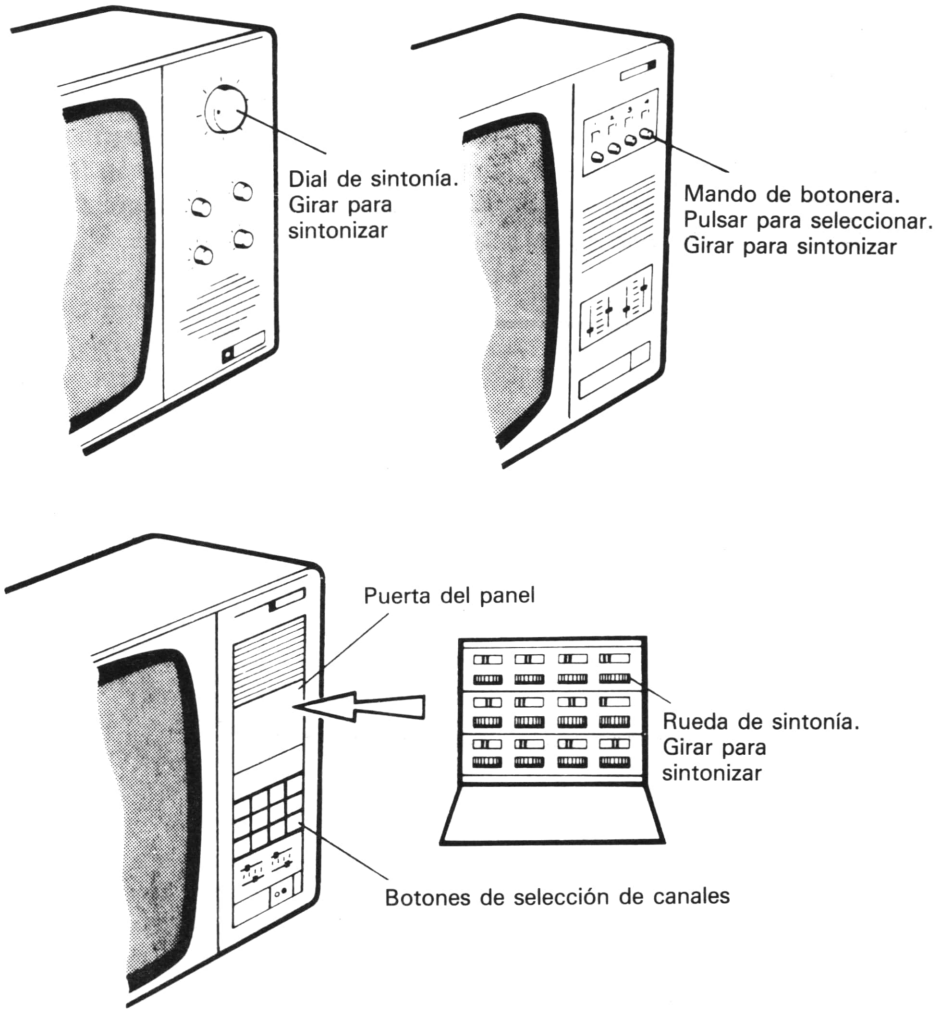

Fig. 1.6 Programa para comprobar las funciones de grabación y lectura del cassette.

Lo primero que va a ver, en el supuesto de que no haya tocado el QL desde que lo encendió, es el texto que se muestra en la figura 1.7. La última línea de este texto es el mensaje de copyright, que en mi ordenador lleva la fecha de 1983. Por encima están las instrucciones para seleccionar monitor o televisor. Esta selección es necesaria porque, si se le indica que se ha conectado un televisor, el QL genera caracteres más grandes. Para ello se pulsa la tecla F2. Cualquiera que sea la tecla que se pulse, F1 o F2, una de las unidades de microdrive se pondrá en marcha durante un instante. Si tiene conectado

un televisor y pulsa F1, verá la pantalla dividida en tres regiones (blanca, roja y negra), pero no las letras que normalmente aparecerían al lado izquierdo de la pantalla. Si se ha equivocado en la selección del modo de visualización, puede elegirlo nuevamente; para ello asegúrese de que no hay ningún cartucho introducido en las ranuras de los microdrives y pulse el botón RESET que se encuentra a la derecha del teclado. En la pantalla aparecerá durante unos instantes un mosaico y luego el mensaje de la figura 1.7.

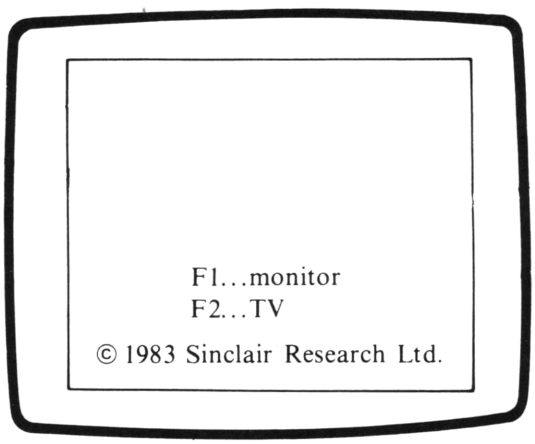

Fig. 1.7 Mensaje que aparece en la pantalla cuando se enciende el ordenador.

Cuando aparezca en la pantalla este mensaje, gire el mando de sintonía muy despacio, a un lado y a otro, hasta que consiga la máxima nitidez posible de las letras. En un televisor de color las letras no van a ser perfectas, pero procure al menos que sean estables y lo más nítidas posible.

Los televisores más antiguos de blanco y negro tienen botones de sintonía mecánicos (Fig. 1.6*b*). Normalmente son cuatro botones, de los cuales hay que dedicar uno, quizá el último, a sintonizar el ordenador. Presione firmemente el botón hasta que encaje y gírelo para sintonizar. Empiece por girarlo completamente en sentido contrario al de las agujas del reloj; no le sorprenda si tiene que darle muchas vueltas antes de que llegue al tope. Ahora gire el botón en sentido contrario hasta que aparezca la imagen. Si no lo consigue en ninguna posición, compruebe si el cable está bien conectado a la entrada de antena. Si ha instalado el adaptador en «Y», compruebe si con otro botón recibe normalmente las señales de TV. Si es así, al televisor no le ocurre nada: intente otra vez sintonizarlo al QL.

Los televisores de color modernos tienen como selector de canales un con-

junto de botones de presión muy pequeños, o bien botones de contacto. Estos botones sirven solamente para la selección de canales, no para sintonizar. La sintonía se realiza mediante unas ruedecitas que se encuentran en un panel, generalmente oculto tras una puerta en un lateral o en la cara frontal del televisor (Fig. 1.6*c*). Los botones suelen estar numerados, y las ruedas están marcadas con el mismo número. Elija la de número más alto (6, 8 o 12) y pulse el botón correspondiente a ese número. Para sintonizar se gira la rueda correspondiente. Al igual que en los demás casos, su objetivo es lograr una imagen lo más nítida posible.

Los televisores de este tipo suelen realizar automáticamente una sintonía fina cuando se cierra la puerta del panel; por lo tanto, ciérrela, pues de lo contrario los circuitos que se encargan de mantener la sintonía no funcionan y usted tendrá que estar continuamente resintonizando. El QL es capaz de dar una imagen de buena calidad en casi todos los televisores. Si en el suyo la imagen es inestable o los colores son borrosos, compruebe la sintonía. Si el problema persiste, haga que revisen su televisor, o cambie de modelo.

MANOS A LA OBRA

Ahora que ya tenemos en la pantalla la imagen enviada por el QL, ha llegado la hora de empezar a conocer el ordenador. En cuanto pulse F1 o F2 para elegir el modo de visualización, verá en la pantalla un cuadrado azul pulsátil: es el *cursor*. Sirve para indicar el lugar de la pantalla en el que aparecerá la letra o signo correspondiente a la siguiente tecla que usted pulse. Cuando lo haga, el cursor se desplazará un lugar a la derecha. Es importante advertir que nada que el usuario teclee puede ser perjudicial para la máquina; lo peor que puede ocurrir es que pierda un programa que esté almacenado en memoria. Sí es posible, en cambio, dañar el QL dejando que le caigan líquidos encima (¡cuidado con el café!) o conectándolo a otros aparatos cuando está encendido. También se pueden dañar los cartuchos de microdrive si se los extrae cuando están funcionando.

Apague *siempre* el ordenador y todos los aparatos a él conectados antes de desconectar alguna de las clavijas que van a la parte posterior del QL. No apague *nunca* el ordenador ni extraiga el cartucho mientras el motor del microdrive esté funcionando, es decir, mientras oiga el ruido del motor o esté encendido el piloto que hay a la izquierda de la ranura del microdrive. Los números de los microdrives no están marcados en la carcasa. El de la izquierda es el número 1 (en las instrucciones, MDV1) y el de la derecha es el 2 (MDV2). Enseguida explicaremos el funcionamiento de los microdrives; antes vamos a ocuparnos de las teclas.

EL TECLADO

Es hora de echar un vistazo al teclado, porque a través de él es como vamos a comunicarnos con el QL. Si dejamos aparte las teclas F1 a F5 del lado izquierdo, observamos que las teclas del QL son muy parecidas a las de una máquina de escribir. La colocación de las letras y los números es igual. Si usted ha utilizado máquinas de escribir, en particular una electrónica, no le será difícil habituarse a este teclado. Cada vez que pulse una de las teclas literales (teclas marcadas con una letra), verá cómo aparece en la pantalla la letra correspondiente. La letra será *minúscula*, no *mayúscula*. Lo mismo que la máquina de escribir, el teclado del QL normalmente produce letras minúsculas. Si lo que quiere es escribir mayúsculas tendrá que pulsar una de las dos teclas SHIFT al mismo tiempo que la de la letra deseada.

Las órdenes se le pueden dar al ordenador tanto en minúsculas como en mayúsculas. Si quiere escribir en mayúsculas permanentemente, pulse la tecla CAPS LOCK. Para volver a minúsculas basta con pulsar nuevamente CAPS LOCK. Lamentablemente, no hay una luz que indique en qué situación se encuentra CAPS LOCK; hay que probar tocando una tecla. Cualquiera que sea la situación de CAPS LOCK, el funcionamiento de las teclas que están marcadas con dos signos no se altera. Al pulsar una de estas teclas sin SHIFT se obtiene el signo que está grabado en la parte inferior de la tecla. Si quiere escribir el signo superior tendrá que usar SHIFT. Por ejemplo, si pulsa la tecla del 8 sola obtendrá el número 8; si la pulsa acompañada de SHIFT obtendrá *.

Además de las teclas normales de máquina de escribir, hay otras teclas especiales. Por ejemplo, a la izquierda del teclado se encuentran las teclas marcadas con ESC (escape) y CTRL (control). El funcionamiento de estas teclas se describe detallamente en el manual; sus nombres no son fiel reflejo de las funciones que realizan. Son útiles en los programas que se suministran con el QL, pero probablemente no tanto en los que usted escriba. No obstante, en este momento es interesante conocer una de las funciones de la tecla CTRL: escriba una palabra, pulse simultáneamente la tecla CTRL y la de «cursor a la izquierda» y verá cómo desaparece la última letra de la palabra. Si las mantiene pulsadas, las demás letras irán desapareciendo también con rapidez, y el cursor se irá moviendo hacia la izquierda. Ésta es una de las formas de corregir los errores mecanográficos. Al pulsar al mismo tiempo la tecla CTRL y la barra espaciadora se interrumpe la ejecución del programa en curso. Cuando se detiene el programa de esta forma, se lo puede volver a poner en marcha, ya que el programa no se borra de la memoria. Otra forma de detener un programa es pulsar el botón RESET, pero entonces el programa se borra de la memoria. Utilizaremos más adelante este método, pues a veces es la forma más fácil de hacer volver el ordenador a su estado

inicial. Las cuatro teclas marcadas con flechas son las que controlan el movimiento del cursor. Las más útiles son las que están a la izquierda de la barra espaciadora; las otras dos están a su derecha. Las primeras son las que se utilizan en la corrección de errores mecanográficos; con ellas se pueden borrar palabras o letras, sustituir una letra por otra o insertar letras donde se desee. Explicaremos estas funciones en el apéndice A. Ya hemos visto el efecto de la tecla «cursor a la izquierda» combinada con la de CTRL. Finalmente las cinco teclas de la izquierda del teclado son las teclas de «función programable». Se las utiliza en los programas suministrados con el ordenador.

La tecla más importante de todas, al menos para nosotros en este momento, es la tecla ENTER. Está en la posición equivalente a la de retorno del carro en las máquinas de escribir electrónicas, pero su acción no es la misma. Al pulsar ENTER se está diciendo al ordenador que se ha terminado de escribir una instrucción o una orden. Si el lector está acostumbrado a manejar una máquina de escribir tendrá que cambiar alguno de sus hábitos. Por ejemplo, en una máquina de escribir tiene que pulsar la tecla de retorno del carro cada vez que llega al final de una línea, para luego continuar escribiendo al principio de la siguiente. La tecla ENTER del ordenador hace bastante más que esto. Si el texto que se está introduciendo es más largo que la longitud de una línea de pantalla, el ordenador se encarga de pasar *automáticamente* a la línea siguiente. La tecla ENTER no se debe utilizar con este propósito, sino solamente cuando se desea que el ordenador almacene una instrucción o ejecute una orden. No obstante, siempre que se pulse ENTER el cursor saltará al principio de la línea siguiente. La posición en la que va a aparecer la siguiente letra o signo es siempre la indicada por el cursor.

Observe que la acción de las letras se repite, y muy deprisa, si las mantiene pulsadas durante algún tiempo. Si escribe algo que el ordenador no comprenda y luego pulsa ENTER, la máquina responderá con un mensaje de error:

bad mane

(nombre incorrecto). Esto ocurre porque un ordenador es una máquina bastante tonta. Sólo entiende unas cuantas palabras, las denominadas *palabras de instrucción* o *palabras reservadas*. Si lo que se le comunica no está entre esas palabras reservadas, o si se las utiliza incorrectamente, para el ordenador el texto introducido es un «nombre incorrecto». Puede tener sentido para las personas, pero no para la máquina.

COMPROBACIÓN DEL MICRODRIVE

El ordenador QL, como todos los demás, almacena las instrucciones y la información de todo tipo en su memoria, pero sólo mientras está encendido. En cuanto lo apague, el contenido de la memoria desaparecerá, y de forma

irrecuperable, aun cuando vuelva a encenderlo inmediatamente. Por esta razón es necesario guardar los programas y toda la información que éstos utilicen en algún otro medio de archivo. Los soportes más utilizados son los discos y las cintas magnéticas. En el momento de escribir este libro se ignora si el QL será conectable «oficialmente» a unidades de disco magnético, aunque se han anunciado dispositivos «extraoficiales» (es decir, ajenos al fabricante) que permitirán tal conexión. Tiene incorporadas, como ya sabemos, dos unidades de microdrive.

El ordenador dispone de circuitos que convierten las instrucciones de un programa en señales eléctricas aptas para ser grabadas en la cinta de los cartuchos de microdrive. Cuando se vuelven a leer estas señales, otros circuitos se encargan de convertirlas en instrucciones de programa. De esta forma, las unidades de microdrive del QL permiten grabar los programas en cinta y su posterior lectura. Antes de abordar el resto del libro, es conveniente que el lector se familiarice con el uso de las unidades de microdrive para grabar y leer programas.

Lo más fácil para empezar es leer (o *cargar*) un programa. El QL viene con cuatro programas grabados en cartuchos de microdrive. Practicaremos con ellos. Como se puede leer en las notas que acompañan los programas, no se debe trabajar directamente con los cartuchos originales. Esto no significa que este medio de grabación no sea fiable; se trata simplemente de una medida de seguridad. Todos los sistemas de grabación están expuestos al riesgo de pérdida de señales; los profesionales de la informática guardan sistemáticamente copias de todo lo que consideran importante. Durante el tiempo que tardé en escribir este libro, dos de mis cartuchos de microdrive se estropearon. El manual explica cómo hacer *copias de seguridad* (*back-up*) de esos programas. No obstante, mientras no se haya adquirido experiencia en la utilización de los microdrives, se corre el riesgo de perder algún programa valioso; por ello aconsejamos al lector que aplace para más adelante la copia de programas. Dicho sea de paso, consideramos absurdo pagar dinero por un programa que no pueda ser copiado. Si el fabricante del programa anuncia orgulloso que su programa está protegido contra copias, sencillamente *no lo compre*.

Vamos a cargar, pues, el programa EASEL. El ordenador debe estar recién inicializado, sin nada en la memoria. En caso de duda, pulse el botón RESET y a continuación la tecla F1 o la F2, dependiendo de si tiene conectado un monitor o un televisor. Saque el cartucho de microdrive de la caja, sujetándolo por los bordes ranurados. Insértelo en la unidad de microdrive de la izquierda (número 1) empujándolo suavemente, con la etiqueta hacia arriba. El cartucho debería entrar suavemente. Si ofrece resistencia, no lo fuerce; compruebe que lo está introduciendo de la forma correcta. Si la resistencia persiste, pruebe con otro cartucho.

Pero no basta con insertar el cartucho para que el programa se cargue.

Antes hay que dar al ordenador las instrucciones oportunas, y aquí es donde vamos a darnos cuenta por primera vez de lo torpe que es el ordenador. Las órdenes que se den al ordenador han de tener la forma precisa, sin el menor error, pues de otro modo no las entenderá. Un detalle que hay que cuidar es el signo de subrayar (_), que se encuentra en la tecla que está a la derecha de la del 0, en la fila superior del teclado. Observe que para obtener este signo hay que pulsar la tecla SHIFT al mismo tiempo, pues de lo contrario lo que se escribe es el signo «menos» (–). Las instrucciones de carga del microdrive no funcionan con el signo –, sino solamente con el de subrayar, _. Por otra parte, si se pulsa la tecla SHIFT al mismo tiempo que la barra espaciadora, no se produce el espacio; las palabras quedan unidas, y éste es un error mecanográfico que el QL no acepta.

Para cargar el programa EASEL hay que teclear, exactamente según se indica, lo siguiente:

LRUN MDV1_BOOT

y luego pulsar ENTER. Según el manual, lo que hay que teclear es EXEC_W MDV1_EASEL, pero esta orden no dió resultado en mi QL. Las órdenes se pueden teclear en mayúsculas o en minúsculas, siempre que el signo de subrayar se escriba correctamente. Si la orden ha sido escrita sin errores, el motor del microdrive se pone en marcha y en la pantalla aparece el mensaje de copyright de Psion. Esto no quiere decir que ya esté cargado el programa; para ello todavía hay que esperar, y bastante más que si el programa estuviera grabado en disco. Hay otra forma de cargar este programa y los otros tres que se suministran con el ordenador: consiste en encender la máquina, insertar el cartucho en la unidad de microdrive número 1 y *después* pulsar F1 o F2. Al pulsar esta tecla empieza automáticamente la carga y ejecución del programa.

Si se pretende utilizar la orden EXEC_W recomendada por el manual, el motor del microdrive se pone en marcha y el ordenador se bloquea. Entonces hay que extraer el cartucho y reinicializar la máquina con RESET. Una vez cargado el programa EASEL, el lector puede consultar en el manual las instrucciones correspondientes a este programa y probarlo por su cuenta. De esta manera se familiarizará con el manejo del ordenador. También comprobará lo difícil que es leer textos escritos en color en la pantalla de un televisor. Ésta es la razón por la que en las aplicaciones profesionales es imprescindible utilizar un monitor.

GRABACIÓN

Después de practicar con las unidades de microdrive, el lector puede probar el procedimiento que el manual sugiere para hacer copias de seguridad. Ob-

sérvese que este procedimiento constituye otro ejemplo de cómo cargar y ejecutar automáticamente un programa sin más que introducir un cartucho en MDV1 después de inicializar el ordenador y antes de pulsar F1 o F2. Ésto es una excepción a la regla según la cual se deben extraer los cartuchos antes de pulsar el botón RESET. Hay que estar *completamente seguros* de que los cartuchos están en las unidades correctas; de lo contrario, puede ocurrir que se copie un cartucho vacío sobre uno que contenga un programa valioso. Ésta es la razón por la que antes sugeríamos que el lector aplazara la copia de programas hasta que tuviera más soltura en el manejo de los microdrives. La siguiente etapa consiste en crear un programa, grabarlo y luego cargarlo. Antes de nada es necesario *inicializar* (o *formatear*) un cartucho vacío.

La operación de inicializar es algo similar a trazar unas pautas en un cuaderno de caligrafía. La cinta de microdrive es un bucle sin fin con un punto de unión en el que no se puede grabar. La máquina tiene que grabar señales a intervalos sobre la cinta para marcar los tramos utilizables. Estas marcas facilitarán más tarde la localización de la información, por lo mismo que resulta más fácil encontrar una palabra en un libro si se sabe en qué página está. Las cintas de microdrive no se pueden utilizar mientras no se las haya inicializado. Ponga un cartucho nuevo en la ranura MVD1, teclee la orden

format mdv1_prueba

y pulse ENTER. La palabra «prueba» es el nombre de identificación que quedará grabado en el cartucho; se puede utilizar cualquier nombre que no tenga más de diez letras (si tiene más de diez, las últimas serán truncadas). El nombre no puede contener signos de puntuación, tales como puntos, comas, etc., aunque sí puede contener el signo de subrayar. Para comprobar que el nombre asignado al cartucho ha quedado grabado en él, teclee

dir mdv1_

y pulse ENTER. El signo de subrayar de esta orden es imprescindible. Si lo omite, el ordenador responderá con el mensaje de error «not found» («no lo he encontrado»).

Para poder probar el sistema de grabación antes hay que tener algún programa que grabar, y para ello tendrá que escribirlo. Es conveniente, para partir de cero, reinicializar el ordenador con el botón RESET, salvo que acabe de encenderlo.

Escriba el número 10 y la palabra «rem». Es indiferente que escriba rem o REM. Compruebe que no se ha equivocado y pulse ENTER. El efecto de esta acción es pasar la línea de instrucción «10 REM» a la memoria del QL. Cuando escribió el primer dígito, el 1 apareció en la pantalla en la posición del cursor. Al pulsar ENTER el cursor ha saltado a la línea siguiente y al

mismo tiempo la instrucción ha aparecido en el extremo superior izquierdo de la pantalla en la forma

10 REMark

Si observa que se equivoca al escribir, corrija con la tecla CTRL y la de «cursor a la izquierda», con lo que el cursor se va desplazando a la vez que borra los caracteres. Una vez borrada la letra incorrecta, escriba la correcta. Cuando haya corregido la línea entera, pulse ENTER. Si después de todo se ha equivocado y ha escrito algo como REN o RWM, el ordenador no le avisará; en realidad, el ordenador acepta cualqueir cosa que se escriba tras un número. Para corregir una línea tal como

20 Ren

puede, sencillamente, escribir la versión correcta:

20 rem

y luego pulsar ENTER. Nótese la diferencia con respecto a la anterior serie de ordenadores ZX. Escriba ahora todas las restantes líneas de la figura 1.8;

```
10 REMark
20 REMark
30 REMark
40 REMark
```

Fig. 1.8 Programa para comprobar las funciones de grabación y lectura del microdrive.

no olvide pulsar ENTER al final de cada línea. Los números se denominan *números de línea* y tienen dos cometidos. Primero, indican al ordenador que lo que les sigue es una línea de programa; segundo, sirven como guía al ordenador, pues éste normalmente ejecuta las instrucciones en el orden de los números de línea. En todo momento puede comprobar cómo va quedando su programa sin más que pedir al ordenador que lo «liste». En informática, *listar* significa hacer que el ordenador escriba una lista con el contenido de parte de su memoria. Puesto que ha escrito los números de línea, puede estar seguro de que el ordenador ha ido guardando las instrucciones en su memoria; si escribe «list» y luego pulsa ENTER, verá el programa en la pantalla. No le sorprenda ver que el ordenador ha convertido las letras minúsculas en mayúsculas (por ejemplo, rem en REM), pues ésta es una de las acciones normales del ordenador, aparte de completar la palabra (REMark, observa-

ción). Compruebe mediante el listado que su programa es idéntico al de la figura 1.8.

Ahora compruebe que el microdrive está preparado, con un cartucho en la ranura MDV1. Escriba lo siguiente:

SAVE MDV1_miprueba

y pulse ENTER. La palabra «SAVE» es la orden que indica al ordenador que queremos almacenar (es decir, grabar) un programa en el cartucho; «mdv1_» significa que estamos utilizando la unidad de microdrive número 1 en esta grabación. La palabra «miprueba» es el *nombre de fichero* por el que el ordenador reconocerá el programa cuando se le pida que lo lea. También en este caso es necesario el signo de subrayar. Es posible grabar un programa sin darle nombre (pero poniendo el signo de subrayar), pero esto no es una buena costumbre. Normalmente se grabarán varios programas en una cinta, y sus nombres respectivos son necesarios para ordenar a la máquina que lea el que se desea cargar. Después de pulsar ENTER se oye el ruido del motor del microdrive, y enseguida reaparece el cursor en la pantalla. Aunque el motor siga funcionando, el programa ya ha quedado grabado, lo que se puede comprobar escribiendo dir mdv1_ (ENTER). El motor del microdrive se para automáticamente al final del proceso.

La cinta de microdrive es muy fina y se mueve mucho más deprisa que las cintas de cassette ordinarias. Por lo tanto, se enreda con cierta facilidad; cuando eso ocurre, el microdrive se para, sin grabar ni leer, y expulsa ligeramente el cartucho. *A veces* se consigue desenredar una cinta golpeándola contra una superficie dura (por ejemplo, la de una mesa).

El principal riesgo para los cartuchos es la contaminación de la cinta, que provocará el mensaje de error «bad medium» (medio de grabación defectuoso). Observe las siguientes precauciones:

1. No toque la cinta con los dedos. Sujete el cartucho solamente por los bordes ranurados.
2. No introduzca nada (bolígrafo, lapicero, etc.) en el cartucho.
3. No encienda ni apague el ordenador sin antes extraer los cartuchos de las ranuras de microcrive.
4. Guarde los cartuchos en sus cajas, en un lugar fresco y seco.
5. Sea cuidadoso al insertar y extraer los cartuchos.
6. Al insertar los cartuchos asegúrese de que han encajado en la ranura antes de poner en marcha el microdrive. En ocasiones es necesario sujetar el cartucho mientras el microdrive funciona.
7. No inserte los cartuchos mientras no sea necesario.

Fig. 1.9 Sugerencias sobre la conservación de los cartuchos de microdrive.

Vamos a comprobar que la grabación ha sido correcta. Escriba NEW (nuevo) y pulse ENTER, con lo que borrará el programa de la memoria. Para evitar confusiones, borre la palabra tecleando CLS y pulsando ENTER. Ahora escriba LIST y pulse ENTER; acaba de pedir al ordenador que liste las instrucciones del programa actual y, por lo que se ve en la pantalla, habrá comprobado que no hay ningún programa. Ahora puede leer las instrucciones de la cinta. Escriba LOAD MDV1_miprueba y pulse ENTER. El motor del microdrive funcionará durante unos instantes y luego se detendrá. Escriba LIST y pulse ENTER. Si todo ha ido bien, en la pantalla aparecerán las instrucciones de su programa. Cuando haya adquirido soltura suficiente para grabar y cargar programas, estará en condiciones de empezar a programar. Como comprobará en el futuro, si ha dedicado una hora a introducir un programa por el teclado, vale la pena dedicar unos minutos a grabarlo para ya no tener que volver a escribirlo. Los cartuchos de microdrive son mucho más caros que las cintas de cassette de otros ordenadores, pero funcionan mejor y no requieren ajustes. De lo único que hay que preocuparse es de conservar correctamente los cartuchos. En la tabla de la figura 1.9 se dan algunas instrucciones a este respecto.

CARGA Y EJECUCIÓN AUTOMÁTICA DE PROGRAMAS

Más adelante, cuando esté escribiendo sus propios programas, el lector puede desear que los programas se carguen y ejecuten inmediatamente al pulsar la tecla F1 o F2. Éste método es muy sencillo. Al pulsar cualquiera de estas dos teclas, la máquina pone en marcha automáticamente el MDV1 en busca de un programa llamado BOOT. Si hay un cartucho en esa ranura y en él está grabado un programa con el nombre BOOT, el ordenador lo carga y ejecuta. El programa BOOT puede ser sencillamente una instrucción para cargar otro programa, o bien el programa que usted utilice con mayor frecuencia. También se puede hacer que BOOT cargue y ejecute otro programa poniendo en él la instrucción LRUN (vea los detalles en el manual después de leer este libro).

2

Manejo de la pantalla

Después de leer el capítulo 1 el lector ya se habrá percatado de que el QL, como casi todos los demás ordenadores, recibe las órdenes del usuario a través del teclado. También habrá observado que el ordenador tiene en cuenta una orden cuando se pulsa ENTER, no antes. Ya hemos utilizado la orden LIST, que hace escribir en la pantalla la lista de las instrucciones de que consta el programa. Hay dos detalles útiles que debe conocer antes de seguir adelante. Uno es que se puede borrar la pantalla escribiendo CLS (o cls) y pulsando ENTER. No obstante, las instrucciones que haya en la parte inferior de la pantalla no se borrarán por este procedimiento. Cuando se haya familiarizado con el teclado querrá utilizar las diversas órdenes de edición; éstas están descritas en el apéndice A.

Hay dos formas de utilizar el ordenador. Una es el denominado *modo directo*; consiste en escribir una orden y pulsar ENTER, con lo cual la máquina obedece la orden inmediatamente. Esto puede ser útil en ocasiones, pero la forma usual de utilizar el ordenador es en *modo de programa*. En modo de programa al ordenador se le suministra un conjunto de instrucciones junto con el orden en que debe ejecutarlas. Tal conjunto de instrucciones es lo que se denomina *programa*. La diferencia es importante, porque las instrucciones de un programa se pueden repetir cuantas veces se desee sin requerir más que un mínimo esfuerzo por parte del usuario. En cambio, para repetir una orden directa hay que volver a escribirla completa y pulsar ENTER. El conjunto de palabras que representan las instrucciones y órdenes, junto con las normas para su correcta utilización, es lo que se denomina *lenguaje de programación*. El QL está dotado de una versión actualizada del lenguaje más utilizado en programación de pequeños ordenadores: el BASIC. Esta palabra está formada por las siglas de «Beginners All-purpose Symbolic Instruction Code» (código de instrucciones simbólicas de uso general para principiantes). Este lenguaje fue desarrollado con objetivos didácticos, pero ha

evolucionado hasta convertirse en uno de los más útiles. La versión de BASIC del QL está enriquecida con numerosas instrucciones que no se incluían en el BASIC original.

Hay una cuestión importante acerca de todas las instrucciones de ordenador, tanto si son órdenes directas como si están incluidas en un programa: tienen que estar escritas de una forma precisa, ajustándose a las normas. Esto es particularmente cierto en el caso del QL; si el lector ha programado antes el ZX-81 o el Spectrum, tendrá que hacer un esfuerzo para acostumbrarse al QL. Ya hemos visto cómo se tiene que utilizar el signo de subrayar en las órdenes relativas al microdrive. A medida que avancemos en este libro irá descubriendo la importancia de los espacios. Casi todas las palabras de que constan las órdenes tienen que ir seguidas de un espacio. Esto no es frecuente, y el lector puede estar acostumbrado a no preocuparse por los espacios si ha utilizado antes otros ordenadores. La reacción usual del QL ante la omisión de un espacio es el mensaje de error «bad name». Los ejemplos de este libro mostrarán en qué casos se puede omitir el espacio; no obstante, no se pierde nada dando por supuesto que siempre es necesario.

Examinemos la diferencia que hay entre una orden directa y una instrucción de programa. Si quiere que el ordenador ejecute, como orden directa, la suma de los números 1.6 y 3.2, debe escribir lo siguiente:

PRINT 1.6 + 3.2

(y, recordémoslo una vez más, pulsar ENTER a continuación). Hay que empezar por PRINT (o print) porque ésta es la única forma de indicarle al ordenador que lo que se quiere es que escriba la respuesta en la pantalla. No reconoce instrucciones tales como «ESCRIBE» o «¿CUÁNTO ES?»; solamente entiende unas cuantas palabras, las *palabras reservadas* o *palabras de instrucción*. PRINT es una de esas palabras. Con el fin de que el lector reconozca más fácilmente las palabras reservadas, a partir de ahora las escribiremos siempre en mayúsculas (aunque, como ya hemos dicho, el ordenador las entiende igual en minúsculas). En los listados, ya sean por pantalla o por impresora, *siempre* saldrán en mayúsculas. Recuerde que tiene que dejar un espacio después de la T de PRINT. En cambio, no es necesario que haya espacios ni antes ni después del signo +.

Cuando se pulsa la tecla ENTER después de escribir PRINT 1.6 + 3.2, en la pantalla aparece la respuesta, 4.8. Esta respuesta no aparece en el mismo lugar en el que escribimos la orden. Si había borrado la pantalla antes de teclear la orden, la respuesta estará en el extremo superior izquierdo de la pantalla. Si hay algún otro texto excrito en esta región principal de la pantalla, la respuesta aparecerá en la línea siguiente. Cuando se trabaja con un televisor en color, el QL borra y pone como fondo de la pantalla un color rojo. Los resultados de las órdenes aparecen en blanco sobre este fondo. En

cambio, los listados de programa se escriben con caracteres blancos sobre fondo rojo. Las órdenes tecleadas aparecen con caracteres verdes sobre fondo negro en la región inferior de la pantalla. Cuando se conecta un monitor la pantalla se divide de forma diferente. Las letras que el usuario teclea siguen apareciendo en el extremo inferior de la pantalla, pero la separación de listados y resultados se hace en dos rectángulos, uno al lado del otro.

Una vez ejecutada una orden directa, el ordenador la da por terminada y la olvida. No ocurre lo mismo con las instrucciones de programa. Cuando se teclea un programa, las instrucciones no van siendo ejecutadas a medida que se pulsa ENTER al final de cada línea, sino que el ordenador las va archivando en su memoria, en espera de que el usuario le pida que las ejecute. Desde luego, el ordenador necesita poder distinguir entre órdenes directas e instrucciones. En BASIC esto se consigue haciendo que todas las líneas de programa empiecen con un número de línea. Además, los números de línea tienen que ser enteros positivos. Ésta es la razón por la cual no se puede esperar que el ordenador entienda una instrucción tal como 5.6 + 3 =, porque considera que el 5 es un número de línea, y el resto no tiene sentido.

Empecemos a programar las operaciones aritméticas: suma, resta, multiplicación y división. Escogemos éstas por ser fáciles de manejar. Por supuesto, los ordenadores no se utilizan normalmente para este tipo de cálculos, pero es necesario saber cómo realizarlos. En la figura 2.1 se muestra un programa de cuatro líneas que escribe en la pantalla el resultado de otras tantas operaciones aritméticas.

```
10 PRINT 5.6+6.8
20 PRINT 9.2-4.7
30 PRINT 3.3*3.9
40 PRINT 7.6/1.4
```

Fig. 2.1 Un programa aritmético de cuatro líneas.

Observemos detenidamente estas instrucciones, porque hay mucho que aprender de ellas. Para empezar, los números de línea no son 1, 2, 3, 4, sino 10, 20, 30, 40. La razón es la siguiente: si después de escrito el programa se da cuenta de que ha olvidado una instrucción entre las líneas 10 y 20, puede sencillamente escribir un número entero cualquiera comprendido entre el 10 y el 20 (por ejemplo, el 11, el 15, etc.) y a continuación la instrucción olvidada. (El orden en que se introduzcan las instrucciones carece de importancia; lo que cuenta es el número de línea de cada una de ellas. El ordenador las

coloca por orden numérico automáticamente.) Si los números de línea hubieran sido 1, 2, 3, 4, no habría hueco entre ellos para intercalar una nueva instrucción. De todas formas, esto no sería un problema insuperable, porque los números de línea se pueden modificar utilizando las órdenes de edición.

El siguiente detalle en que debe fijarse es la barra que atraviesa el número 0. Esto sirve para distinguirlo de la letra O. El ordenador no acepta el 0 en lugar de la O, ni viceversa. La diferente representación gráfica en la pantalla facilita al usuario la distinción entre estos dos caracteres. Observe que en el teclado el cero no tiene barra, aunque si forma diferente que la O. Practique hasta familiarizarse con estos dos signos.

Otras dos cuestiones importantes. La estrella o, mejor dicho, el asterisco de la línea 30 es el símbolo que el QL utiliza como signo de multiplicar. Así pues, no se puede usar como signo de multiplicar ni x ni X, pues estos caracteres son letras. El signo de división es la barra inclinada (/), no los dos puntos (:); la barra a que nos referimos es la que está en la misma tecla que el signo de interrogación, *no* la de la tecla que está por encima y a la derecha de la de ENTER.

De momento, es de esperar que todo vaya quedando claro. El programa se introduce en el ordenador escribiéndolo tal como está en la figura 2.1. No hace falta dejar espacio entre el número de línea y la P de PRINT, porque el QL lo pondrá automáticamente cuando liste el programa. Pero sí es necesario dejar espacio después de PRINT. El QL, a diferencia de otros ordenadores, es muy intolerante en estos detalles. Cada vez que termine de escribir una línea de programa tiene que pulsar ENTER, antes de escribir el número de la siguiente. El resultado final debe ser exactamente como se ve en la figura. Una vez terminada la introducción del programa, éste se encuentra almacenado en la memoria del ordenador, en forma de un conjunto de códigos numéricos. Hay dos cosas interesantes que aprender ahora; primero, cómo comprobar que el programa efectivamente ha quedado archivado en la memoria; segundo, cómo hacer que la máquina lo ejecute.

Para el primer problema la solución está en la orden LIST. Si lo desea, puede borrar la pantalla con la orden CLS. Escriba LIST y pulse ENTER. El ordenador escribirá en la pantalla el listado de las instrucciones. Observe cómo ha insertado espacios a la derecha de los números de línea. Para ejecutar el programa necesitamos otra orden: RUN. Escriba RUN y pulse ENTER, y verá cómo el ordenador ejecuta el programa. Más concretamente, lo que verá es lo siguiente:

```
12.4
4.5
12.87
5.428571
```

Esta última línea puede darle una idea de la precisión con la que el QL realiza estos cálculos. Una calculadora daría varios decimales más, pero una precisión de seis decimales es suficiente para casi todas las aplicaciones. Habrá observado, si listó el programa antes de ejecutarlo, que los resultados han sido escritos sobre el listado. Para evitarlo, borre la pantalla con CLS ante de teclear RUN.

Cuando después de la instrucción PRINT se escribe una operación aritmética tal como 2.8 * 4.4, lo que el ordenador imprime al ejecutar el programa es el *resultado* del cálculo; es decir, no imprime 2.8 * 4.4, sino el producto.

Esto es útil, pero los resultados escritos en la pantalla pueden no significar nada si se ha olvidado cuáles eran las preguntas. El QL permite escribir cualquier cosa en la pantalla, exactamente como se desee, utilizando las denominadas *cadenas literales.*

```
10 PRINT"2+2=";2+2
20 PRINT"2.5*3.5=";2.5*3.5
30 PRINT"9.4-2.2=";9.4-2.2
40 PRINT"27.6/2.2=";27.6/2.2
```

Fig. 2.2 Utilización de las comillas. Todo lo que se escribe entre las comillas aparece en la pantalla sin modificación.

En la figura 2.2 se da un ejemplo. En cada una de las cuatro líneas, parte del texto está entre comillas. Los signos de punto y coma son *muy* importantes; en otros ordenadores se los puede omitir, pero *no* en éste. Teclee este programa y ejecútelo. Observe el resultado. Todo lo que en el programa estaba entre comillas ha aparecido exactamente como lo había escrito. Las operaciones que estaban fuera de las comillas han sido calculadas, y en la pantalla aparecen los resultados. La combinación de los dos efectos da

2 + 2 = 4

Pero esto no tiene nada de mágico. Si escribe la siguiente línea

15 PRINT "2 + 2 = ";5 * 1.5

y ejecuta el programa, obtendrá sencillamente

2 + 2 = 7.5

El ordenador no hace más que lo que se le manda. Sólo quien no los conozca puede temer que algún día los ordenadores lleguen a dominar el mundo.

Éste es buen momento para insitir en algo que ya hemos dicho. La línea

n.º 15 que añadió al programa ha sido intercalada entre la 10 y la 20. Si no lo cree, liste el programa. Cualquiera que sea el orden en que introduzca las instrucciones, el ordenador las entenderá en orden numérico ascendente.

Con todos los conocimientos que hemos adquirido, ya estamos en condiciones de abordar otros métodos de visualización en la pantalla. La instrucción PRINT, utilizada como hasta ahora, solamente escribe en la pantalla. Para activar la impresora hay una variante que consiste en la palabra PRINT seguida del signo # y de un número (por ejemplo, PRINT #3,). Estas instrucciones solamente tienen utilidad si se dispone de una impresora. En el apéndice B se explica cómo utilizar una impresora con interfaz en serie.

```
10 PRINT"Este es"
20 PRINT"el excelente"
30 PRINT"ordenador QL."
```

Fig. 2.3 Utilización de la instrucción PRINT. Cada instrucción PRINT selecciona una nueva línea en la pantalla.

Pruebe ahora el programa de la figura 2.3. Puede introducir las líneas en el orden que quiera, para demostrarse a sí mismo que en el listado siempre saldrán ordenadas. Cuando borre la pantalla y ejecute el programa, las palabras aparecerán en tres líneas distintas. Esto ocurre porque la instrucción PRINT no se limita a escribir lo especificado, sino que también provoca el salto al principio de la línea siguiente. Dicho sea de paso, observe también que cuando el texto llega a la última línea de la zona de visualización, todas las líneas anteriores se desplazan hacia arriba y la primera desaparece. Este efecto se denomina *scrolling* («rodadura»). Así es como la máquina se las arregla para visualizar muchas líneas en una pantalla en la que sólo caben 24 (aparte de las líneas de la zona inferior de la pantalla, en las que se escriben las órdenes). En el capítulo 8 veremos una orden que provoca el *scrolling*.

No siempre es deseable que cada instrucción PRINT provoque el salto a la línea siguiente. Para evitarlo podemos utilizar signos de puntuación que actúan como modificadores. Antes de pasar al siguiente ejemplo prepárese a adquirir un nuevo hábito: escriba NEW y pulse ENTER. Esto sirve para borrar el programa anterior; borre también la pantalla, si lo desea, con CLS.

Si no utiliza la orden NEW, cabe la posibilidad de que queden líneas del programa antiguo mezcladas con las del nuevo. Cada vez que se escribe una línea de programa, se anula la línea que hasta ese momento tenía ese mismo número, si había alguna. Ahora bien, si en el programa antiguo hay líneas

con números que no se utilicen en el nuevo, esas líneas no se borrarán, y quedarán mezcladas en el nuevo programa. Por ejemplo, en el programa de la figura 2.2 habíamos intercalado la línea 15. Esta línea ha permanecido en la memoria a pesar de que después hemos introducido otras líneas con los números 10 y 20. Si está utilizando una impresora, tenga en cuenta que las instrucciones del tipo PRINT #3 dejan de funcionar como consecuencia de la orden NEW; por lo tanto, tendrá que volver a abrir el canal de la impresora con la orden OPEN #3, SER1.

```
10 PRINT"Este es ";
20 PRINT"el excelente ";
30 PRINT"ordenador QL."
```

Fig. 2.4 Efecto del punto y coma.

Pruebe el programa de la figura 2.4. Hay una diferencia fundamental entre éste y el de la figura 2.3, que comprobará cuando lo ejecute. El efecto del punto y coma que sigue al cierre de las comillas es inhibir el salto a la línea siguiente. Al ejecutar el programa verá que todas las palabras aparecen en la misma línea de la pantalla. En este caso habría sido más fácil incluir todo el texto en una sola línea de programa:

10 PRINT "Este es el excelente ordenador QL"

pero hay ocasiones en que es necesario utilizar el punto y coma para forzar la escritura de dos cadenas literales en una misma línea. Veremos ejemplos de esto más adelante.

FILAS Y COLUMNAS

La visualización elegante del texto consiste en colocar adecuadamente los números y las palabras en filas y columnas de la pantalla. Vamos a dedicar algún tiempo a estas cuestiones. Para empezar, ya sabemos que la instrucción PRINT, en ausencia de modificadores, provoca el salto a la línea siguiente, de forma que el efecto de esta instrucción en el programa de la figura 2.5 no debería sorprendernos. Sin embargo, las líneas 10 y 20 contienen una novedad: dos instrucciones en cada línea. Las instrucciones están separadas por el signo de dos puntos (:). Siempre que se desee se pueden escribir varias instrucciones en una línea de programa, aun cuando ocupen varias líneas físicas en la pantalla. Una limitación práctica es la dificultad de leer una

línea con demasiadas instrucciones. En una línea «multi-instrucción» de este tipo, el QL trata las instrucciones de izquierda a derecha. La instrucción CLS tampoco debe sorprendernos; borra la pantalla y hace que la escritura comience en el extremo superior izquierdo. El efecto es similar al que ejerce cuando se la utiliza como orden directa, pero aquí es automático por estar incluida en el programa.

```
10 CLS:PRINT"Este es el QL"
20 PRINT:PRINT
30 PRINT"dispuesto a trabajar para usted."
```

Fig. 2.5 Utilización de CLS para borrar la pantalla y de PRINT para saltar y borrar líneas.

Otro detalle que se debe observar en el programa de la figura 2.5 es que la línea 20 hace que las dos líneas de texto queden separadas entre sí. En ella hay dos instrucciones PRINT, sin nada que escribir; lo único que tienen que hacer estas instrucciones es pasar a la línea siguiente, borrándola al mismo tiempo. Hay otras formas de conseguir lo mismo, como veremos, pero ésta es la más cómoda.

El programa de la figura 2.6 ilustra el manejo de las columnas. La línea 10 es una instrucción PRINT que actúa con los números 1 y 2. Cuando aparecen en la pantalla, están situados como si ésta estuviera dividida en colum-

```
10 PRINT 1,2
20 PRINT 1,2,3,4
30 PRINT"UNO","DOS"
40 PRINT"UNO","DOS","TRES","CUATRO"
50 PRINT"ESTO YA ES MAS LARGO","DOS"
60 PRINT 1,2,3,4,5,6
```

Fig. 2.6 La coma sirve para escribir en columnas.

nas. El modificador que produce este efecto es la coma; su acción es completamente automática. En el teclado la coma está a la derecha de la M; si se utiliza el apóstrofe que está en la tecla del 7, el efecto no es el mismo. Estos dos signos son parecidos en el teclado, pero completamente diferentes en la pantalla. En mi QL el apóstrofe provoca un mensaje de error, a pesar de que

el manual provisional asegura que su efecto es saltar a la línea siguiente. Como demuestra la línea 20, hay cuatro columnas en la pantalla, cada una de ocho caracteres. Si se intenta escribir algo en la quinta columna, el cursor salta a la primera columna de la línea siguiente. La coma sirve tanto para números como para palabras, como demuestran las líneas 30 y 40. Cuando escriba instrucciones de este tipo, recuerde que las comas tienen que quedar *fuera* de las comillas. Las comas que estén dentro de las comillas serán transcritas sin modificación, y no afectarán a la posición del texto en la pantalla. Observe que si se intenta escribir en una columna un texto que no quepa en ella, parte del texto pasará a la columna siguiente. La línea 50 ilustra este efecto: la primera frase ocupa las tres primeras columnas; la palabra DOS pasa a la cuarta columna. La línea 60 demuestra qué ocurre si se escriben más cosas separadas por comas; los dos últimos números pasan a la línea siguiente.

Las comas son útiles si lo que se pretende es utilizar cuatro columnas de igual anchura. Pero hay un método mucho más versátil, que se programa con la palabra reservada AT (en) y que permite escribir palabras en la pantalla en cualquier posición y en cualquier orden. Normalmente, la instrucción PRINT hace que el ordenador pase a la línea siguiente y no permite retroceder a la anterior. Utilizando AT se puede escribir en cualquier sitio, e incluso reemplazar una línea con otra.

Si el lector ha programado el Spectrum, más vale que olvide cómo utilizaba en él la instrucción AT. En el QL la instrucción AT tiene que preceder a la instrucción PRINT. Puede estar en la línea inmediatamente anterior a la de PRINT, o incluso en la misma línea, separadas por el signo de dos puntos (:). AT *no se pone después de* PRINT, como ocurre en el Spectrum y otros ordenadores.

AT va seguida de dos números (parámetros). El primero es el número «de columna». En cada línea de pantalla del QL caben 37 caracteres (letras, números o signos de puntuación). Cada uno de estos caracteres ocupa una columna; podemos utilizar un número para caracterizar cada posición en horizontal dentro de una línea. El número varía entre 0 (izquierda de la pantalla) y 36 (derecha). El segundo parámetro de AT es el número de fila, que va de 0 (línea superior de la pantalla) a 19 (última línea de la zona de visualización de la pantalla).

Es conveniente ilustrar esta instrucción con un ejemplo. Observe el programa de la figura 2.7. La línea 10 borra la pantalla. La línea 20 introduce la primera instrucción AT; aquí 0,0 significa que la escritura se debe realizar en el extremo superior izquierdo de la pantalla. Es imprescindible dejar un espacio entre la T de AT y el primer número. La A de ARRIBA IZQUIERDA aparecerá en el rincón superior izquierdo, como en todo caso ocurriría después de CLS. La siguiente línea no es más que una pérdida de tiempo:

```
10 CLS
20 AT 0,0:PRINT"ARRIBA IZQUIERDA"
30 PAUSE 100
40 AT 24,19:PRINT"ABAJO DERECHA"
50 PAUSE 100
60 AT 0,19:PRINT"ABAJO IZQUIERDA"
70 PAUSE 100
80 AT 23,0:PRINT"ARRIBA DERECHA"
90 PAUSE 100
100 AT 16,9:PRINT"CENTRO"
```

Fig. 2.7 Cómo utilizar AT para escribir en cualquier lugar de la pantalla.

la palabra PAUSE seguida de un número provoca un retardo. El número representa cincuentavos de segundo, de modo que el número 50 introduce un retardo de un segundo, el 100 de dos segundos, etc.

La siguiente instrucción AT va seguida de 24, 19, de modo que la última A de ABAJO DERECHA queda en el extremo inferior derecho de la pantalla. ¿Cómo hemos calculado estos números? Muy sencillo: la última A de DERECHA tiene que estar en la columna 36; así vamos contando hacia atrás, 35, 34, . . . , hasta llegar a la A de ABAJO, que estará en la posición 24. Como el texto ha de estar en la última línea, el segundo parámetro tiene que ser 19. Con estas explicaciones usted puede continuar el análisis del programa. Los retardos sirven para permitirle observar el orden en que el programa va escribiendo los textos.

En este ejemplo la palabra CENTRO ha quedado centrada en la línea. En la figura 2.8 se indica cómo calcular el número de columna que se debe poner en la instrucción AT para centrar textos. Se cuentan los caracteres de la frase que se desea centrar (entendiendo por «caracteres» tanto letras como dígitos, espacios o signos de puntuación). Este número se resta de 38 y el resultado se divide por 2. El número que se obtiene así (despreciando la parte decimal, si la hay) es el que se debe poner como primer parámetro de AT.

1. Cuente el número de caracteres de la palabra o frase que quiera centrar.
2. Reste ese número de 38.
3. Divida el resultado por dos, ignorando el resto.
4. Utilice el número resultante como parámetro de AT. (Para que el texto quede perfectamente centrado en el televisor puede ser necesario sumar o restar 1 a este número.)

Fig. 2.8 Fórmula para centrar textos.

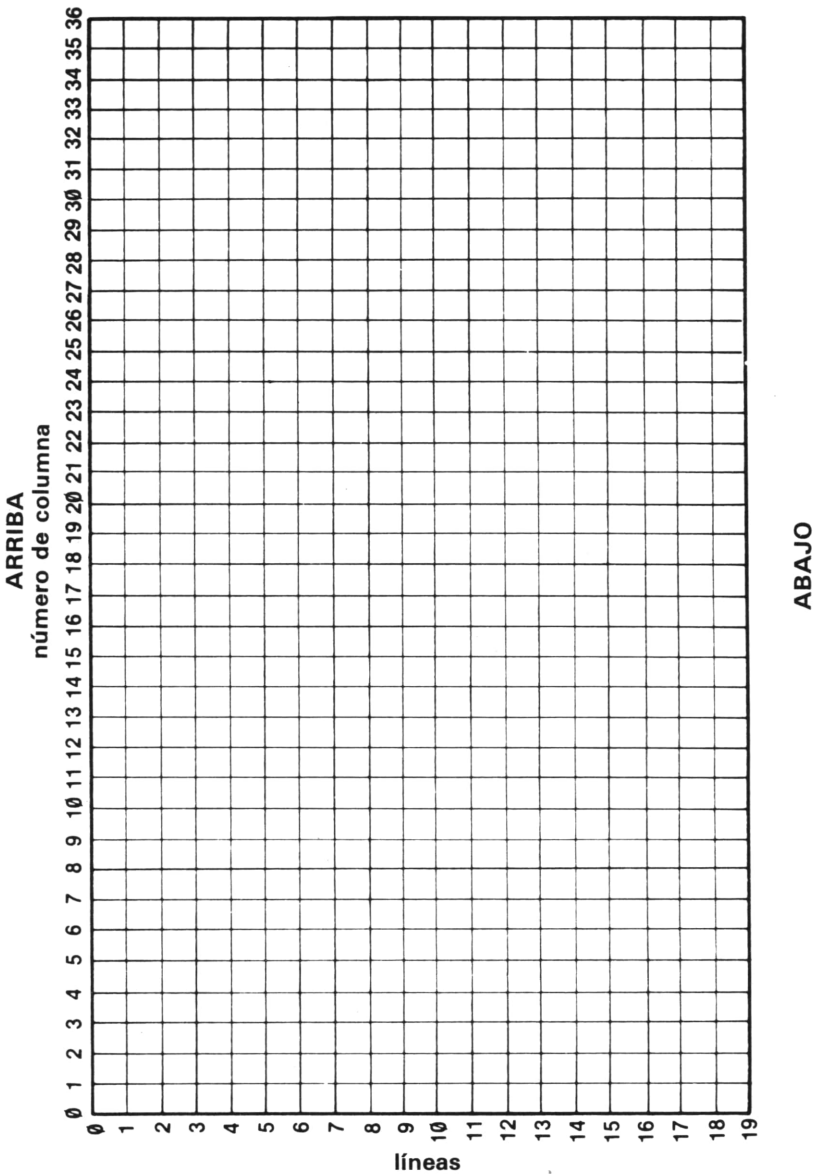

Fig. 2.9 Mapa de los parámetros de la instrucción AT.

En este ejemplo también queríamos centrar en altura. Al ser impar el número de líneas de texto (en este caso, una) no hay forma de centrarlo con precisión; la línea 9 es demasiado alta, la 10 demasiado baja. Más adelante veremos que podemos poner letras en lugar de números en la instrucción AT (y en otras). Esto nos permitirá centrar textos sin el engorro de tener que contar las letras.

En la figura 2.9 damos un mapa de la pantalla, muy útil para la colocación de textos en ella.

Veamos otros recursos para mejorar la presentación de textos en la pantalla. El lector observará cuando empiece a programar por su cuenta que la presentación de palabras en la pantalla dejan mucho que desear. En concreto, observará que las palabras se cortan al final de las líneas de pantalla de forma muy antiestética. El QL ofrece un remedio: la barra «inclinada hacia la izquierda» (\), que está en la tecla del extremo superior derecho del teclado. Cada vez que se ponga este carácter después de una instrucción PRINT, provocará el paso a la línea siguiente. Esto permite organizar un texto largo en una sola instrucción PRINT. En efecto, basta con dividir el texto en grupos de palabras cuya longitud total no sea superior a 37 caracteres, poner cada grupo entre comillas y separar los grupos con \. Para no tener que contar las letras de cada grupo, vigile que el signo de cerrar comillas en una línea de la pantalla *no quede a la derecha* del de abrir comillas en la línea anterior.

```
10 CLS
20 CSIZE 3,1
30 AT 11,2:PRINT"TITULO"
40 CSIZE 2,0
50 AT 2,8:PRINT"Este programa ilustra como se"\"
controla el tamano de los caracteres"
```

Fig. 2.10 Modificación del tamaño de los caracteres con la instrucción CSIZE.

Otro recurso muy interesante del QL es la posibilidad de controlar el tamaño de los caracteres, mediante la instrucción CSIZE. Esta palabra debe ir seguida de dos parámetros: el primero especifica la anchura de los caracteres; el segundo, la altura. Como ejemplo veamos el programa de la figura 2.10. La línea 10 borra la pantalla. La línea 20 establece un nuevo tamaño de caracteres: el máximo posible, útil para títulos. En este tamaño la anchura es 1/3 más grande que la anchura normal en el televisor (8/3 más grande que la anchura de los caracteres en el monitor); la altura es doble de la normal. La diferencia de anchura nos ha obligado a poner menor número de columna en la instrucción 30. Ahora estamos trabajando con 28 caracteres por línea y sólo 10 líneas de altura. La línea 30 escribe la palabra TITULO en caracteres grandes. La línea 40 restablece el tamaño normal de los caracteres. Los parámetros de AT en la línea 50 han sido elegidos para que el texto no se superponga al título. La barra inclinada hacia la izquierda controla el salto a la línea siguiente.

```
10 CLS
20 CSIZE 0,0:PRINT"TAMANO 0,0"
30 CSIZE 1,0:PRINT"TAMANO 1,0"
40 CSIZE 2,0:PRINT"TAMANO 2,0"
50 CSIZE 3,0:PRINT"TAMANO 3,0"
60 CSIZE 0,1:PRINT"TAMANO 0,1"
70 CSIZE 1,1:PRINT"TAMANO 1,1"
80 CSIZE 2,1:PRINT"TAMANO 2,1"
90 CSIZE 3,1:PRINT"TAMANO 3,1"
```

Fig. 2.11 Demostración de los posibles tamaños de los caracteres.

El parámetro de altura de los caracteres en CSIZE sólo puede ser 0 o 1. El de anchura puede variar entre 0 y 3. La figura 2.11 ilustra las diferentes posibilidades. En la pantalla del televisor normalmente sólo se visualizan dos de las cuatro anchuras posibles. Si quiere verlas todas, teclee MODE 512 (ENTER) y ejecute el programa. Para volver al modo normal de visualización en el televisor teclee MODE 256 (ENTER). Para los que utilizamos un televisor las opciones son: 0 o 2 para caracteres estrechos y 1 o 3 para caracteres anchos. El cuanto a la altura, 0 representa la menor y 1 la mayor.

3

Festival de variables

Hasta ahora nuestro trabajo se ha limitado a escribir números y palabras en la pantalla. Ese es uno de nuestros principales objetivos, desde luego, pero también debemos dedicar cierta atención a algunos de los procesos que tienen lugar antes de que los textos puedan aparecer en la pantalla. Uno de ellos es la *asignación* o definición de variables. Consideremos el programa

```
10 CLS
20 LET X=23
30 PRINT"2 POR ";X;" ES ";2*X
40 X=5
50 PRINT"X ES AHORA ";X
60 PRINT"EL DOBLE DE ";X;" ES ";2*X
```

Fig. 3.1 Definición de variables. La letra X representa un número.

de la figura 3.1. Escríbalo, ejecútelo y compare lo que ha escrito con lo que aparece en la pantalla. La primera línea que contiene una instrucción de escritura es la 30. Su efecto es que en la pantalla aparezca

2 POR 23 ES 46

y sin embargo en la línea 30 no hemos escrito los números 23 ni 46. Esto se debe a que la letra X es una especie de código que representa, en este caso, el número 23. Un código de este tipo es lo que denominamos *variable* o *nombre de variable*.

La línea 20 define la variable X y le asigna el valor 23. Esto significa que siempre que utilicemos X (sin encerrarla entre comillas) el ordenador opera-

rá con el número 23. Puesto que X es un solo carácter y 23 tiene dos dígitos, hemos conseguido un ahorro de memoria. El ahorro sería más notable si hubiéramos definido, por ejemplo, X = 2174.3256. La línea 30 demuestra que efectivamente para el ordenador la letra X es el número 23: cada vez que se encuentra la X fuera de comillas, escribe 23, y en lugar de la expresión 2 * X escribe 46. Pero la X no tiene que seguir teniendo el mismo valor para siempre. De hecho, en la línea 40 le asignamos un valor distinto, el 5; las líneas 50 y 60 demuestran que el ordenador ha entendido el cambio.

Ésta es la razón por la que la X se llama «variable»: podemos hacerla variar a voluntad. No obstante, mientras no la cambiemos conservará el valor original. Incluso después de ejecutar el programa anterior, y siempre que no haya añadido o suprimido líneas, puede escribir PRINT X (ENTER) y verá el valor de X en la pantalla. En el listado se puede observar que la asignación se puede realizar de dos formas: con y sin LET. Puesto que el QL las entiende exactamente igual, nos ahorraremos tres letras y un espacio si omitimos LET.

Como nombre de las variables se puede poner una palabra en lugar de una sola letra. La única condición es que debe empezar por una letra, mayúscula o minúscula; los restantes caracteres pueden ser otras letras o dígitos, pero no espacios, ni signos aritméticos (+, −, *, /) ni signos de puntuación. Así pues, son válidos nombres tales como EL_RESTO, apellido, TOTAL y R2D2: a cada uno de ellos se les puede asignar valores distintos. Tenga cuidado con el siguiente detalle: el QL, como otros ordenadores, *no distingue mayúsculas de minúsculas.* Si asigna a la variable pan el número 45 y más tarde a la variable PAN el número 88, tanto pan como PAN tendrán el mismo valor (el último asignado), 88 en este caso. Las palabras reservadas (PRINT, NEXT, RUN, LIST, etc.) no pueden ser nombres de variables (de intentarlo, el ordenador emitiría un mensaje de error). Algunos ordenadores no permiten ni siquiera que los nombres de las variables *contengan* palabras reservadas, de forma que, por ejemplo, ANALISTA no puede ser el nombre de una variable. El QL es más tolerante; se limita a exigir que los nombres de las variables no sean idénticos a ninguna palabra reservada.

También se pueden utilizar nombres para representar palabras y frases (o sea, cadenas literales). La diferencia con respecto a las variables numéricas es que al final del nombre de una variable literal hay que poner el signo de dólar ($). Si N es el nombre de una variable numérica, N$ (que se lee «ene-dólar») es el de una variable literal. El ordenador las trata como perfectamente distintas e independientes que son. La forma de definirlas también es diferente. Cuando se asigna un valor a una variable numérica, el número no se escribe entre comillas. Cuando se define una variable literal, la cadena a la que se iguala sí se escribe entre comillas. Más adelante estudiaremos otros métodos de asignación.

CADENAS LITERALES

En el programa de la figura 3.2 se utilizan *variables literales*, es decir, variables cuyo «valor» es una sucesión de caracteres. En los líneas 10 y 20 se definen las variables; en las líneas 30 a 50 se muestra cómo se utilizan los nombres. Observe que se pueden mezclar los nombres de las variables, sin comillas a sus lados, con texto ordinario, que sí tiene que estar entre comillas. Hay que tener cuidado con esto porque puede resultar confuso. Observe en las líneas 30 a 50 cómo hemos dejado espacios entre las palabras. Los signos de punto y coma son *imprescindibles* para mezclar texto de esta forma.

```
10 CLS:NOMBRE$="QL"
20 PRIMERO$="El excelente":ULTIMO$="trabajando"
30 PRINT PRIMERO$;" ";NOMBRE$;" ";ULTIMO$
40 PRINT"Aqui esta el ";NOMBRE$
50 PRINT PRIMERO$;" ";NOMBRE$;" en accion"
```

Fig. 3.2 Utilización de variables literales. Sus nombres terminan siempre en el signo de dólar.

En la figura 3.3 se muestra otro ejemplo, esta vez asignado a las variables UNA$ y OTRA$ frases largas. Si solamente fuésemos a escribir el texto una vez, no tendría mucho sentido definir estas variables. En cambio, si una frase aparece varias veces en un programa, asignarla a una variable ahorra tener que mecanografiarla repetidamente.

```
10 CLS:UNA$="El nuevo ordenador "
20 OTRA$="que ofrece informatica seria a precio
moderado"
30 PRINT"El QL:"
40 PRINT UNA$!OTRA$
50 PRINT UNA$;" ";OTRA$
```

Fig. 3.3 Asignación de frases de varias palabras a variables literales.

Obsérvese el signo de admiración de la línea 40. Esto es una característica del QL, un «separador inteligente». Cuando se utiliza el signo de admiración como separador de dos variables, la escritura en la pantalla se realizará tratando de evitar que se partan palabras. Si la cadena OTRA$ se escribiera in-

mediatamente después de UNA$, la palabra «informática» quedaría partida (línea 50). Por supuesto, si la cadena total es más larga que la anchura de línea, la inclusión del signo de admiración no evitará la partición de palabras.

MÁS SOBRE CADENAS

Puesto que todo nombre de variable literal ha de terminar en $, una variable tal como A$ no puede ser confundida con la variable numérica A. De hecho, se pueden utilizar ambos nombres en un mismo programa sin provocar confusión (al menos para el ordenador). En la figura 3.4 se ilustra la diferencia. En las dos primeras líneas se definen las variables numéricas A y B y las variables literales A$ y B$. Cuando las escribimos mediante las líneas 40 y 50, no se aprecia diferencia entre A y A$, ni entre B y B$. El QL, a diferencia de los demás ordenadores, es capaz de manejar los números tanto en forma

```
10 A=2:B=3
20 A$="2":B$="3"
30 CLS
40 PRINT A;" por ";B;" es ";A*B
50 PRINT"Los numeros son ";A$;" y ";B$
60 PRINT A$;" por ";B$;" es ";A$*B$
70 REMark IMPOSIBLE EN OTROS ORDENADORES - DARIA
N MENSAJE DE ERROR"
80 C$="AZUL":D$="VIOLETA"
90 PRINT C$*D$
100 REMark ESTO YA NO ES POSIBLE
```

Fig. 3.4 Las variables literales y numéricas se parecen en este listado, pero para el ordenador son diferentes.

de variable numérica como en forma de variable literal, según demuestra la línea 60. La diferencia entre los dos tipos de variables se hace bien patente, en cambio, cuando el ordenador trata de ejecutar la línea 90; puede multiplicar variables numéricas, o dos variables literales que representen números, pero no puede multiplicar variables literales cualesquiera. Si se intenta realizar una operación imposible, como la de la línea 90, el programa se detiene con el mensaje de error «At line 90 error in expression» (en la línea 90, error en una expresión). Más adelante veremos que hay ciertas operaciones que son específicas de las cadenas y que, por lo tanto, no pueden ser realizadas

con números; si se intenta tal cosa, el mensaje de error que se obtiene es el mismo. La razón de esta especificidad de las operaciones radica en la distinta forma en que el ordenador almacena los dos tipos de variables, que está diseñada para facilitar a la máquina las operaciones que deba realizar con unas y otras.

En el QL, el signo + *siempre* realiza la suma, si ésta es posible. La operación de unir cadenas, denominada *concatenación*, se realiza con el signo &. La figura 3.5 ilustra la diferencia entre concatenación y suma (líneas 60 y 70). La concatenación proporciona un medio para generar cadenas compuestas ahorrando trabajo de mecanografía. Consideremos el programa de la figura 3.6. En la línea 10 se define las variables A$ y B$ como una especie de «adornos» para utilizar en un título. El título en sí se define en la línea 20. La línea 40 escribe la cadena final.

```
10 A$="Matesanz"
20 B$="Villanueva"
30 CLS
40 PRINT"Llameme sencillamente ";"A$&"-"&B$;" ,
dijo."
50 PRINT:A$="123":B$="456"
60 PRINT"La cadena global es ";A$&B$
70 PRINT"La suma es ";A$+B$
```

Fig. 3.5 Concatenación o unión de cadenas, operación distinta de la suma aritmética.

```
10 A$="***":B$="###"
20 s$="EL NUEVO QL"
30 CLS
40 PRINT A$&B$&s$&B$&A$
```

Fig. 3.6 Utilización de la concatenación para construir un título con orla.

```
10 CLS
20 A$=FILL$("*",10)
30 B$=FILL$("#",5)
40 AT 1,2:PRINT A$&B$&"TITULO"&B$&A$
```

Fig. 3.7 Ejemplo de utilización de FILL$ para formar una cadena de caracteres repetidos.

Relacionada con la concatenación, hay una instrucción que sirve para formar una cadena consistente en un determinado carácter repetido cuantas veces se desee. Se trata de una instrucción FILL$, que debe ir seguida de dos parámetros entre paréntesis. El primero es el carácter que se debe repetir; el segundo es el número de repeticiones deseadas. Por ejemplo, si se define G$ = FILL$("$",20), la cadena G$ será una sucesión de 20 signos de dólar. En la figura 3.7 se da un ejemplo de aplicación de esta instrucción.

ENTRADAS

Hasta ahora, cada vez que hemos querido visualizar texto en la pantalla hemos tenido que incluirlo previamente en el programa. Pero no tenemos que resignarnos a esta limitación, ya que el ordenador nos permite introducir datos, tanto numéricos como literales, en el programa durante la ejecución de éste. Una operación de este tipo es lo que se denomina «entrada». La instrucción de BASIC que realiza esta función es INPUT.

```
10 CLS
20 PRINT"Escriba su nombre"
30 INPUT NOMBRE$
40 CLS:PRINT:PRINT
50 PRINT"Buenos dias, ";NOMBRE$
```

Fig. 3.8 Utilización de la instrucción INPUT. La instrucción 50 escribe en la pantalla el nombre tecleado.

En la figura 3.8 se da un ejemplo de utilización de esta instrucción. Como el programador no conoce de antemano el nombre del usuario, no podía incluirlo explícitamente en el programa. Al ejecutar este programa, en la pantalla aparece lo siguiente:

Escriba su nombre

En la línea siguiente a este mensaje se puede ver el cursor (azul parpadeante). Esto indica que el ordenador está esperando que se escriba algo y a continuación se pulse ENTER. Mientras tanto, el programa estará detenido en la línea 30. Al teclear el nombre no es necesario poner comillas; escríbalo tal como quiera verlo luego escrito en la pantalla. Al pulsar ENTER, el nombre que ha tecleado es asignado a la varible NOMBRE$. Ahora se reanuda la ejecución del programa; la línea 40 borra la pantalla y avanza la posición de escritura a la tercera línea de la pantalla. La línea 50 escribe un texto y

su nombre a continuación. Por supuesto, en lugar de su nombre podría haber tecleado Pato Donald o Capitán Trueno o cualquier otra cosa. El ordenador no tiene forma de saber si se le está diciendo la verdad. Incluso aunque no teclee nada, y se limite a pulsar ENTER, el programa continuará, sin asignar ninguna cadena a la variable NOMBRE$.

La instrucción INPUT sirve también para la introducción de datos numéricos. En la figura 3.9 se da un ejemplo. Este programa es similar al anterior.

```
10 CLS:PRINT"Escriba un numero, por favor"
20 INPUT n
30 CLS:PRINT
40 PRINT n;" por 2 es ";2*n
```

Fig. 3.9 INPUT aplicada a una variable numérica. El dato que se introduzca tiene que ser un número.

Cuando el programa se detiene y el cursor aparece en la pantalla, el ordenador espera que se teclee un número y se pulse ENTER. La acción de pulsar ENTER hace que el número tecleado sea asignado a la variable numérica N. La línea 40 prueba que el ordenador ha captado el número introducido. Cuando se utiliza INPUT seguida del nombre de una variable numérica, lo que se teclee antes de pulsar ENTER tiene necesariamente que ser un número. Si alguno de los caracteres tecleados no es un dígito, el ordenador se niega a aceptar el dato y emite un mensaje de error: «at line 20 error in expression». En cambio, si después de INPUT se escribe el nombre de una variable literal, el ordenador aceptará *cualquier cosa* que se teclee; el error se producirá más tarde si se intenta realizar operaciones aritméticas con una cadena que no esté compuesta íntegramente por dígitos.

```
10 CLS
20 INPUT"Por favor, escriba su nombre ";NOMBRE$
30 PRINT
40 PRINT"Encantado de conocerle, ";NOMBRE$
```

Fig. 3.10 Utilización de INPUT para escribir una frase que solicita el dato.

La instrucción INPUT se puede utilizar de forma que parezca que el ordenador está atento a lo que el usuario teclea. En la figura 3.10 se ilustra otra forma de utilizar INPUT. La frase de solicitud del dato está entre comillas,

seguida de un punto y coma y del nombre de la variable. La línea 20 de este programa equivale a las dos líneas siguientes:

```
15 PRINT "POR FAVOR, ESCRIBA SU NOMBRE";
20 INPUT NOMBRE$
```

En este caso el cursor aparece en la *misma línea* que la frase. La respuesta del usuario también estará en la misma línea, a menos que sea tan larga que no quepa en ella y tenga que continuar en la siguiente. En una instrucción como ésta se puede poner una coma o un signo de admiración en lugar del punto y coma. Pruébelo. El efecto de la coma es hacer que la respuesta se visualice en la siguiente columna de tabulación. En cambio, el apóstrofe no ejerce efecto aparente en mi QL. Como ya sabemos, la barra inclinada hacia la izquierda (\) fuerza el salto a la línea siguiente en las instrucciones PRINT; el mismo efecto se produce si se la pone en lugar del punto y coma en la instrucción INPUT. Una variante de la instrucción INPUT consiste en especificar el «número de canal». Ya ha observado el lector que la pantalla está dividida en cuatro zonas. Las instrucciones PRINT e INPUT se pueden referir a cada una de esas zonas si se especifica cuál de ellas se desea. Por ejemplo, si en la figura 3.10 se pone INPUT #0, (atención a la coma) en lugar de PRINT, el mensaje y la respuesta serán enviados a la zona inferior de la pantalla. INPUT #2, tiene el mismo efecto que INPUT, pero INPUT #3, hace que se utilice la zona normalmente reservada a los listados de programas. En mi QL estas versiones de INPUT provocan también la aparición de una línea parpadeante en la pantalla. Otras variaciones han ocasionado la pérdida del control de la máquina.

```
10 CLS
20 INPUT"Por favor, nombre y numero: ";NM$,N
30 PRINT:PRINT
40 PRINT"El nombre es ";NM$
50 PRINT"El numero es ";N
```

Fig. 3.11 Dos variables en una instrucción INPUT.

La instrucción INPUT puede ir seguida de varios nombres de variables, y se pueden mezclar variables literales con variables numéricas en una misma instrucción. En el programa de la figura 3.11 se captan dos variables una literal y otra numérica en una instrucción INPUT. Al llegar a la instrucción 20, el ordenador escribe la frase y espera hasta que el usuario introduzca los dos datos. Sólo hay una forma correcta de hacer esto: escribir un nombre y pulsar ENTER. El cursor salta a la línea siguiente, lo que indica que

sigue esperando otro dato; entonces se teclea el número y se pulsa ENTER. Las líneas 40 y 50 escribirán en la pantalla el nombre y el número introducidos. Si en la línea 20 se pone el signo de admiración en lugar de la coma, los dos datos se pueden escribir en la misma línea.

```
10 CLS
20 INPUT"Por favor, cuatro numeros: ";A,B,C,D
30 PRINT
40 PRINT"Su suma es ";A+B+C+D
```

Fig. 3.12 Instrucción INPUT que capta cuatro números.

Veamos otro ejemplo. El programa de la figura 3.12 solicita cuatro números. También en este caso, los números son asignados a variables numéricas. La línea 40 escribe su suma.

LECTURA DE DATOS

Hay otra forma de introducir datos en un programa durante su ejecución. Consiste en leer los datos de una lista y utiliza dos instrucciones: READ (leer) y DATA (datos). La instrucción READ, cada vez que se ejecuta, hace que el programa lea un elemento de la lista. El programa identifica la lista porque ésta va precedida de la palabra DATA. Los elementos de la lista tienen que estar separados por comas. Cada vez que se lee un elemento, el «puntero» pasa a señalar el elemento siguiente, el cual queda dispuesto para ser leído por la siguiente instrucción READ.

En el capítulo 5 volveremos a ocuparnos de estas dos instrucciones. Ahora vamos a ver un ejemplo sencillo, el de la figura 3.13. La línea 20 lee un dato, el primero de la lista (línea 80) y lo asigna a la variable J. La línea 30 lo escribe; el punto y coma del final impide que el cursor salte a la línea siguiente

```
10 CLS
20 READ J
30 PRINT"El periferico ";J;
40 PRINT" es un ";
50 READ N$
60 PRINT N$
80 DATA 5,"lector de disco"
```

Fig. 3.13 Utilización de READ y DATA para suministrar datos a un programa.

de la pantalla. El punto y coma de la línea 40 tiene el mismo efecto. La línea 50 lee el nombre que figura como segundo elemento de la lista y lo asigna a la variable literal N$. La línea 60 lo escribe. La línea 80 contiene la lista de datos. Obsérvese que en esta línea el número está sin comillas y la cadena literal con ellas. Ésta es precisamente la norma.

Siempre hay que ser extremadamente cuidadoso con la congruencia entre los datos de las listas de DATA y las variables de READ. Si una instrucción READ requiere un dato numérico (por ejemplo, READ A), el elemento de la lista que va a ser leído tiene que ser un número. En cambio, si la variable de la instrucción READ es literal (por ejemplo, READ A$), la naturaleza del elemento leído es indiferente (es decir, no importa que sea literal o numérico). Recuerde que si se lee un número con READ A$, el QL permite realizar con él operaciones aritméticas.

Hay otra instrucción, RESTORE, relacionada con la pareja READ/DATA. Si se ejecuta por segunda vez el programa de la figura 3.13, se produce el siguiente mensaje de error:

At line 20 end of file (fin de fichero en la línea 20)

La razón es la siguiente: cuando se ejecutó por primera vez, se leyeron todos los datos de la línea DATA; en la segunda ejecución ya no hay más datos que leer, y el ordenador informa de ello. Hay dos soluciones para este problema. La primera consiste en teclear CLEAR (ENTER) antes de RUN. La segunda, en incluir en el programa la línea

5 RESTORE

RESTORE (restaurar) significa para la máquina «ir otra vez al principio de la línea DATA». Más adelante volveremos a encontrarnos esta instrucción, pues tiene otras aplicaciones. De momento, no olvide poner RESTORE al principio de todo programa que contenga las instrucciones READ ... DATA.

La utilidad de READ ... DATA se hace más patente cuando se trata de leer una larga lista de datos. Los datos que se deben leer por este procedimiento son los que el programa utilice en todas las ejecuciones, no los que el usuario deba introducir en el teclado.

NÚMEROS

El trabajo que hemos desarrollado hasta ahora habrá convencido al lector de que los ordenadores hacen bastante más que cálculos numéricos. Pero, por supuesto, hay aplicaciones en las que la posibilidad de manejar números es fundamental. Si el lector piensa utilizar su ordenador en aplicaciones cien-

tíficas o de ingeniería, la capacidad de la máquina para realizar calculos es mucho más importante que si pensara dedicarlo a proceso de textos, o incluso a contabilidad. Es tiempo, pues, de dar un repaso a las posibilidades de cálculo del QL. Será un repaso somero, porque no tenemos espacio para analizar detalladamente todas las funciones matemáticas de que esta máquina está dotada. En general, si el lector sabe en qué consisten funciones tales como «sen» o «exp», no tendrá ninguna dificultad a la hora de incluir estas funciones y operaciones en sus programas. En caso contrario, puede perfectamente omitir la lectura de los apartados que traten estas cuestiones.

La operación más sencilla y elemental es la de contar. Nociones relacionadas con ella son las de *incrementar* (contar hacia arriba) y *decrementar* (contar hacia abajo). Incrementar un número es sumarle 1. Decrementar un número es restarle 1. En BASIC estas operaciones se programan de una forma que puede parecer confusa, como muestra el programa de la figura 3.14. La línea 20 asigna a X el valor 5; la línea 30 lo escribe. La línea 40 incrementa X; la extraña instrucción X = X + 1 hace que el nuevo valor de la variable sea igual al que tenía antes más 1. El resto del programa demuestra que la operación de incrementar X ha sido correctamente realizada.

```
10 CLS
20 X=5
30 PRINT"El valor de X es ";X
40 X=X+1:PRINT
50 PRINT"Ahora hemos hecho X=X+1":PRINT
60 PRINT"El valor de X es ";X
```

Fig. 3.14 Operación de incrementar, utilizando el signo = con el significado de «se convierte en».

El uso del signo = con el sentido de «se convierte en» es algo a lo que el lector tendrá que ir acostumbrándose. También podríamos escribir la instrucción

$$X = X - 1$$

cuyo efecto sería decrementar X, es decir, hacer X igual a su antiguo valor menos 1. También podríamos utilizar X = X * 2 para duplicar X, o X = X/3 para dar a X un valor igual a la tercera parte del que tenía antes. En la figura 3.15 se modifica X por multiplicación y suma.

En la figura 3.16 se ilustra la utilización de varias funciones matemáticas. Una función es una instrucción que actúa sobre un número para producir otro número. La línea 10 asigna el valor 2.5 a X. La línea 20 escribe el cua-

```
10 CLS
20 X=5:PRINT"X es ";X
30 PRINT
40 X=2*X+4
50 PRINT"Ha cambiado!"
60 PRINT"X es ahora ";X
```

Fig. 3.15 Modificación de una variable mediante una *expresión algebraica.*

```
10 CLS:X=2.5
20 PRINT"X elevado al cuadrado es ";X^2
30 PRINT
40 PRINT"Su raiz cuadrada es ";SQRT(X)
50 PRINT
60 PRINT"Su logaritmo natural es ";LN(X)
70 PRINT"y su logaritmo decimal es ";LOG10(X)
```

Fig. 3.16 Algunas funciones matemáticas.

ABS(X) Convierte los números negativos en positivos.
ACOT(X) Da el ángulo (en radianes) cuya contangente es X.
ATN(X) Da el ángulo (en radianes) cuya tangente es X.
COS(X) Da el coseno del ángulo X (radianes).
COT(X) Da la cotangente del ángulo X (radianes).
DEG(X) Convierte el ángulo X, dado en radianes, en grados.
DIV Realiza la división entera (da la parte entera del cociente).
EXP(X) Da el valor del número *e* elevado a X.
INT(X) Da la parte entera de X.
LN(X) Da el logaritmo natural de X.
LOG10(X) Da el logaritmo decimal de X.
MOD Da el resto de la división entera.
PI Da el número π, relación de la longitud al diámetro de la circunferencia.
RAD(X) Convierte el ángulo X, dado en grados, en radianes.
RANDOMISE Inicializa una nueva sucesión de números aleatorios.
RND Da un número aleatorio comprendido entre 0 y 1.
RND(X TO Y) Da un entero aleatorio comprendido entre X e Y, ambos inclusive.
SIN(X) Da el seno del ángulo X (radianes).
SQRT(X) Da la raíz cuadrada de X.
TAN(X) Da el valor de la tangente del ángulo X (radianes).

Fig. 3.17 Funciones numéricas del QL. (No se preocupe si no sabe para qué sirve alguna de ellas; si es así, seguramente no las necesita.)

drado de X, esto es, X multiplicado por X. Esto se programa escribiendo X↑2; el carácter que el QL utiliza para esta función está en la tecla del «6». Para obtener la raíz cuadrada de un número utilizamos la instrucción SQRT(). Otra forma posible es X↑.5, pero SQRT(X) es más fácil de escribir (y de recordar, para los usuarios de habla inglesa). Para otras raíces se utilizan potencias fraccionarias; por ejemplo, la raíz cúbica es X↑(1/3). LN(X) da el logaritmo natural o neperiano. El logaritmo decimal se obtiene con LOG10(X).

En la figura 3.17 se da una relación de las funciones matemáticas del QL, junto con una breve descripción de cada una de ellas. Algunas de estas funciones sólo tendrán utilidad en aplicaciones científicas, técnicas o estadísticas. Sin embargo, otras serán necesarias en las situaciones más inesperadas, como por ejemplo en la programación de gráficos.

PRECISIÓN EN LOS CÁLCULOS

Uno de los problemas de los ordenadores pequeños es la precisión con la que realizan los cálculos. Como el lector sabe, la fracción 1/3 no se puede expresar con exactitud mediante ningún número decimal. La aproximación a su valor verdadero será tanto mejor cuanto mayor sea el número de dígitos con que lo escribamos. Así, 0.33 es más preciso que 0.3, y 0.333 es aún más preciso. El ordenador convierte casi todos los números con los que trabaja en una fracción y un multiplicador. La fracción no es decimal, sino de una forma especial denominada *fracción binaria.* Esta conversión raramente es exacta, y es particularmente torpe para números tales como 1, 10, 100, ... y 0.1, 0.01, 0.001, ..., es decir, para las potencias de 10. Para no tener que escribir cosas como 3 − 2 = 0.9999999, el ordenador redondea los números de este tipo, hacia arriba o hacia abajo, antes de escribirlos. No todos los

```
10 CLS
20 PRINT 1/3,2/3
30 PRINT 1/11,10/11
40 PRINT 1/3+2/3,1/11+10/11
```

Fig. 3.18 Ilustración de cómo maneja el QL los números «difíciles».

ordenadores son tan hábiles en este aspecto como el QL. La figura 3.18 muestra cómo maneja el QL fracciones tales como 1/3, 2/3, 1/11 y 10/11. Los números que escribe en la pantalla han sido redondeados a 7 decimales,

pero internamente tiene que almacenarlos con precisión mucho mayor. El redondeo se efectúa de forma que la suma de fracciones dé el resultado correcto.

Por la forma en la que el ordenador escribe la fracción 1/11, el lector puede observar que hay otra manera de escribir los números. Si ha trabajado en algún área en la que se utilicen números muy grandes o muy pequeños (física, química, ingeniería), esto le será familiar. Por si no lo ha visto antes, esta forma de escribir los números se llama forma estándar o *notación científica.* Un número en notación científica consiste en un número comprendido entre 0 y 10 multiplicado por una potencia entera de 10. Consideremos, por ejemplo, el número 132000. Si desplazamos el punto decimal cinco lugares hacia la *izquierda* (lo que equivale a *dividir* por 10 elevado a 5), el número resultante es 1.32. Para volver al valor original, tendremos que multiplicar este número por 10 elevado a 5, o sea, por E5, pues ésta es la forma en que los ordenadores escriben las potencias de 10. En resumen, el número 132000 se puede escribir en la forma 1.32E5.

Probemos ahora con un número pequeño: 0.00036. Este número es igual a 3.6 por 10 elevado a menos 4, o sea, 3.6E-4; el signo «menos» aparece porque hemos tenido que desplazar el punto decimal cuatro lugares hacia *la derecha.* El QL, como casi todos los ordenadores pequeños, acepta y escribe números en esta notación. La conversión es automática. El QL escribe los números en notación científica solamente cuando le es necesario, es decir, cuando en notación ordinaria requerirían más de siete dígitos.

La mayor parte de los ordenadores permite el manejo, mucho más preciso y económico en lo que a ocupación de memoria se refiere, de una clase especial de números: son las llamadas *variables enteras.* Para el QL, un *entero* es un número entero comprendido entre −32768 y +32767. El nombre de una variable entera ha de terminar con el signo %. Al asignar valores a varibles enteras, es necesario que el valor cumpla las limitaciones mencionadas. Por ejemplo, si intentamos hacer algo como

A% = 32800

provocaremos el mensaje de error «Overflow» (rebosamiento o sobrepasamiento). El programa de la figura 3.19 muestra cómo se deshace el ordenador de la parte decimal cuando el resultado de los cálculos con números enteros no es entero. En efecto, en la línea 40 se asigna a la variable entera X% el valor de la variable X, que es 3.7. Sin embargo, la línea 50 imprime 4. La parte decimal 0.7 ha sido redondeada hacia arriba, a 1. Esto es infrecuente, y afortunado, porque muchos ordenadores, en lugar de redondear, se limitan a tomar la parte entera; es decir, omiten 0.7 para dar 3.

La ventaja de utilizar variables enteras es doble. La primera es que los cálculos aritméticos (salvo la división) dan resultados exactos, sin necesidad de

```
10 CLS:X=3.7
20 PRINT"X es un numero ordinario igual a ";X
30 PRINT" X% es un entero."
40 X%=X
50 PRINT"El valor de X% es ";X%
60 Y%=7.5
70 PRINT"7.5 es ";Y%;" considerado como entero."
```

Fig. 3.19 Utilización de variables enteras.

redondeo. La división es una excepción, como demuestra la línea 70 del programa de la figura 3.19. En segundo lugar, los enteros ocupan menor espacio en la memoria, y la velocidad de cálculo es mucho mayor con variables enteras que con cualesquiera otras. El QL permite realizar la división entera utilizando las funciones DIV y MOD. DIV da la parte entera del cociente. Por ejemplo, si tecleamos

PRINT 7 DIV 3

el resultado que aparece en la pantalla es 2, porque, efectivamente, 7/3 es 2 más una parte decimal. La operación MOD da el resto de la división entera. Si tecleamos

PRINT 7 MOD 3

el resultado es 1, ya que 1 es el *resto* de dividir 7 por 3. Estas dos operaciones tienen su utilidad en los métodos de programación más avanzados, para los que no tenemos espacio en este libro.

Otra posibilidad especialmente útil en las aplicaciones matemáticas es la *definición de funciones.* El interés de la definición de funciones reside en que, con sólo teclearlas una vez, se las puede utilizar en el programa tantas veces como se desee. La figura 3.20 ilustra cómo funciona la instrucción DEF FN. El ejemplo es muy sencillo, y en la práctica nunca se definiría una

```
10 CLS
20 PRINT"Por favor, introduzca tres numeros "
30 INPUT A,B,C
40 PRINT"La suma es ";sum(A,B,C)
100 STOP
1000 DEFine FuNction sum(A,B,C)
1010 RETurn A+B+C
1030 END DEFine
```

Fig. 3.20 Definición de funciones.

función solamente para esto. Las líneas 20 y 30 captan tres números. En la línea 40 se escribe el valor de un ente llamado sum(A,B,C); para que el ordenador conozca esta función, tenemos que definirla en algún lugar del programa. La primera línea de la definición debe empezar con DEF FN (el ordenador se encarga de escribir las palabras completas, DEFine FuNction). A continuación, se pone el nombre elegido para la función. La siguiente línea empieza con RETURN, y en ella se pone la «fórmula» de la función. En este caso, la suma de los números que hemos asignado a las variables A, B y C. En este ejemplo, la línea de END DEF no es necesaria. Veamos ahora el ejemplo de la figura 3.21, para el que hemos seguido más al pie de la letra las instrucciones del manual. En este caso la definición va al principio del

```
10 DEFine FuNction hipot(a,b)
20 LOCal lado
30 lado=SQRT(a^2+b^2)
40 RETurn lado
50 END DEFine
60 CLS
70 INPUT"Los lados de un triangulo son ";x,y
80 PRINT"La hipotenusa es ";hipot(x,y)
```

Fig. 3.21 Otro ejemplo de función definida, esta vez más útil.

programa; definimos la magnitud «lado» como variable local. Una variable *local* es una magnitud que se utiliza en cierta zona del programa y carece de valor en el resto. Aquí queremos que «lado» sea el valor que la definición entregue al resto del programa, y por eso en la línea 40 especificamos que «lado» es la magnitud que debe ser entregada. Para comprobar que esta variable es local, ejecute el programa e intente escribirla con una orden PRINT: obtendrá un asterisco, pero no un valor numérico. Observe también que en las líneas 70 y 80 hemos utilizado las variables x e y, mientras que las que intervienen en la definición son a y b. En realidad, DEF FN indica al ordenador qué debe hacer con un par de números; el nombre que se dé a esos números carece de importancia. Los recursos de este tipo facilitan la labor de programación; a medida que vaya adquiriendo soltura aprenderá a utilizarlos con fluidez.

4

Repetición de tareas

Una de las actividades para las que los ordenadores están especialmente bien dotados es la repetición de un conjunto de instrucciones. El QL no es excepción; de hecho, maneja más instrucciones «de repetición» que los demás ordenadores que conozco. Describiremos en primer lugar la más sencilla de estas instrucciones: REPEAT (repetir). Sí, ya sé que el lector experto estará pensando que más sencilla aún es GOTO. Lo que ocurre con GOTO es que ha pasado de moda, porque son más los problemas que crea que los que resuelve. Esta instrucción ha sido abandonada por las dos máquinas que incorporan una versión realmente moderna de BASIC, y una de ellas es el QL.

REPEAT significa exactamente lo que la palabra sugiere: repetir la acción que se va a especificar a continuación. Normalmente, los programas ejecutan las instrucciones en orden ascendente de los números de línea. La instrucción REPEAT puede alterar este orden, de forma que una o varias líneas puedan ser ejecutadas repetidamente. Para que esto ocurra necesitamos una línea que contenga la palabra REPEAT y el nombre que queramos dar a la rutina; a continuación vendrán las líneas que indican al ordenador cuáles son las tareas que debe repetir. El grupo de instrucciones termina con END REP y el nombre de la rutina.

En la figura 4.1 se da un ejemplo de un sencillo mecanismo de repetición o *bucle*. La línea 10 contiene la instrucción «REPEAT tarea». En esta línea

```
10 REPeat tarea
20 PRINT"Este es el QL"
30 END REPeat tarea
```

Fig. 4.1 Un bucle infinito muy sencillo. Para detenerlo, pulse simultáneamente la tecla CTRL y la barra espaciadora.

«tarea» es el nombre de referencia que damos a lo que se va a repetir. Se puede utilizar cualquier nombre, a excepción de palabras reservadas o de nombres de variables cuyo valor se vaya a utilizar más tarde. La línea 20 hace que se escriba en la pantalla la frase «Este es el QL». En este sencillo ejemplo, esto es todo lo que queremos repetir. La línea 30 da por terminada la acción de repetir, mencionando otra vez el nombre de la rutina. Si no se pone el nombre, se provoca un mensaje de error. Pero en este grupo de instrucciones no hay nada que indique al ordenador cuándo debe dejar de escribir la frase, por lo que en la pantalla aparecerá una y otra vez

Este es el QL

hasta que pulse CTRL y la barra espaciadora para «interrumpir el bucle». Siempre que la máquina parezca haber entrado en un bucle infinito se la puede parar de esta forma; el programa se detiene, pero no completamente. Si a continuación se teclea CONTINUE y se pulsa ENTER, el programa se reanuda a partir de la situación en la que fue interrumpido. Si lo que se desea es detener definitivamente el programa, por ejemplo para modificarlo o para grabarlo, se debe teclear CLEAR (ENTER). Hacer esto es muy importante si se van a introducir modificaciones, pues en caso contrario la máquina se puede bloquear más tarde, cuando menos se espere. Observe que cuando se pulsa la barra espaciadora con CTRL para interrumpir un bucle, la máquina emite un mensaje curioso: «In line 20, not complete» (no completo en la línea 20). El objeto de este mensaje es recordar al usuario que debe teclear CONTINUE o CLEAR.

```
10 CLS:N=10
20 REPeat bucle
30 PRINT N
40 N=N+1
50 END REPeat bucle
```

Fig. 4.2 Bucle que realiza un contador muy rápido.

Probemos ahora un bucle en cuyo interior se desarrolla una actividad algo más interesante. El programa de la figura 4.2 escribe en la pantalla un número cada vez que recorre el bucle, o sea, con cada *pasada por el bucle*. La línea 10 asigna el valor inicial 10 a la variable N. En la línea 20 empieza la rutina de repetición, que contiene dos líneas de instrucciones. La primera es la 30, PRINT N; la segunda es la 40, N = N + 1. La línea 50 marca el final de la rutina. El efecto global es visualizar en la pantalla un contador muy rápido. También este bucle es infinito, y para interrumpirlo habrá que pulsar la ba-

rra espaciadora con CTRL. Al hacerlo se tiene la oportunidad de comprobar cómo está funcionando el programa. Teclee CLEAR si quiere abandonar el programa definitivamente.

Los bucles infinitos de este tipo no tienen demasiado interés; REPEAT permite crear bucles mucho más constructivos. Para ello necesitamos que el programa pueda detener el bucle cuando interese, es decir, cuando se satisfaga cierta condición prescrita. Por ejemplo, podríamos querer interrumpir el bucle cuando la variable que se utiliza como contador alcance el valor 100, o cuando se teclee ULTIMO en un programa que capte nombres. Las instrucciones que necesitamos ahora son IF ... THEN y EXIT.

IF tiene que ir seguido de una condición. Por ejemplo, IF N = 10 o IF NOMBRE$ = "ULTIMO". Después de la condición se puede poner la palabra THEN (entonces). Las líneas siguientes pueden contener otras instrucciones. Así, para interrumpir el bucle cuando la condición sea verdadera, se utilizará la instrucción EXIT seguida del nombre de la rutina. Como siempre, el QL es muy exigente en cuanto a los nombres que han de seguir a estas instrucciones y en cuanto a los espacios entre instrucciones y nombres. También se puede omitir THEN, siempre que no se ponga nada en su lugar. Como veremos más adelante, entre el principio de la comprobación, IF, y el final, END IF, se pueden poner varias condiciones y ocupar para ello cuantas líneas se necesiten. El final de REPEAT se programa como de costumbre.

```
10 CLS
20 PRINT"Introduzca nombres"
30 PRINT"(ULTIMO para terminar)"
40 REPeat nombres
50 INPUT nombre$
60 IF nombre$="ULTIMO" THEN
70 EXIT nombres
80 END IF
90 END REPeat nombres
100 PRINT"FIN DEL PROGRAMA"
```

Fig. 4.3 Detección de un finalizador en un bucle.

Si todo esto le parece complicado, considere el ejemplo de la figura 4.3. Las líneas 10 a 30 escriben las instrucciones para el usuario, el cual puede introducir nombres hasta que decida terminar el programa, para lo cual deberá teclear ULTIMO. ¿Cómo podemos programar este tipo de control? Empezamos en la línea 40 definiendo la rutina de repetición, a la que llama-

mos «nombres». La acción que se debe repetir es la captación de nombres por el teclado: de ello se encarga la línea 50. Un programa «real» tendría que hacer algo con esos nombres, pero nosotros lo simplificamos para poder resaltar lo que nos interesa en este momento. Una vez introducido un nombre y pulsada la tecla ENTER, tenemos que determinar si ese nombre es ULTIMO. Esta comprobación la realiza la línea 60. ¿Qué ocurre si se cumple la condición, es decir, si el nombre es ULTIMO? Pues que se ejecuta la línea 70, EXIT nombres, con lo cual se ha interrumpido el bucle. No es necesario teclear CLEAR cuando se interrumpe un bucle por este procedimiento; la máquina se da por enterada automáticamente. Como no necesitamos más comprobaciones, en la línea 80 tenemos el END IF (aquí no hay que poner ningún nombre de rutina). Finalmente, la línea 90 contiene END REPEAT nombres, lo que señala el final del bucle. La línea 100 escribe un mensaje para informar de que el programa ha salido del bucle.

Cuando ejecute el programa, podrá comprobar que éste se detiene cuando se introduce «ULTIMO», pero no «Ultimo» ni «último». Podemos modificar el programa en previsión de esta posibilidad recurriendo a la operación lógica OR. Si hacemos que la línea 60 sea

```
60 IF nombre$ = "ULTIMO" OR nombre$ = "Ultimo" OR nombre$ =
"último" THEN
```

habremos incluido tres condiciones en lugar de una; basta con que se cumpla una cualquiera de ellas para que el programa ejecute las líneas que siguen a THEN. Se puede utilizar la operación AND para combinar condiciones; por ejemplo, si queremos terminar el programa cuando el valor de un contador haya llegado a 100 *y al mismo tiempo* el nombre tecleado sea ULTIMO, podemos programar

```
60 IF nombre$ = "ULTIMO" AND N = 100 THEN
```

(donde hemos supuesto que N es la variable del contador).

ELSE

La instrucción IF ... THEN es una fórmula de comprobación que se puede utilizar combinada con una rutina REPEAT o independientemente de ella. Pero existe una versión de IF ... THEN más potente. Puede ir seguida de ELSE (si no) para realizar una nueva comprobación y tomar la decisión pertinente. Aclaremos esto con el ejemplo de la figura 4.4.

Es un sencillo programa que simula el lanzamiento de una moneda. Las líneas 10 a 30 realizan el trabajo habitual. En la línea 40 empieza la rutina de repetición. La línea 50 es la fundamental para la simulación. RND sig-

nifica «elegir al azar»; cuando esta instrucción va seguida de dos números entre paréntesis, el resultado es un número entero comprendido entre esos dos números (ambos inclusive). En este ejemplo los números son 1 y 2, de forma que cada vez que se ejecuta esta línea la máquina elige aleatoriamente un 1 o un 2 y lo asigna a la variable X. La línea 60 pide al usuario que

```
10 CLS
20 PRINT"CARA O CRUZ"
30 PRINT"Teclee F para terminar"
40 REPeat juego
50 X=RND(1,2)
60 INPUT"Cara (C) o cruz (X)? ";res$
70 IF res$="F" THEN
80 EXIT juego
90 END IF
100 IF res$="C" AND X=1 OR res$="X" AND X=2 THEN
110 PRINT"Usted gana"
120 ELSE PRINT"Usted pierde"
130 END IF
140 REN REPeat juego
150 PRINT"FIN DEL JUEGO"
```

Fig. 4.4 Ampliación de la instrucción IF con ELSE.

teclee C o X, para cara o cruz. La letra tecleada es asignada a la variable res$. Ahora tenemos que hacer una comprobación con res$. Lo primero que tenemos que comprobar es si debemos interrumpir el bucle, es decir, si el usuario ha tecleado «F». Esta comprobación la realizan las líneas 70 a 90. En la línea 80 hemos puesto EXIT porque queremos abandonar el programa si el usuario ha tecleado «F». Las otras comprobaciones se realizan en las líneas 100 a 130. La línea 100 determina si el usuario ha acertado; para ello hacemos que X = 1 signifique «cara» y X = 2 «cruz». La línea 110 escribe «Usted gana» y el bucle se repite. La línea 120 entra en acción cuando el usuario no ha acertado; escribe «Usted pierde» y deja que se repita el ciclo. Solamente cuando el usuario teclee «F» llegará el programa a la línea 150.

La importancia de ELSE reside en que da una opción. Si se cumple la condición, se realiza cierta acción (por ejemplo, escribir un mensaje); si no se cumple, se realiza una acción distinta. Más adelante veremos cómo programar comprobaciones más complejas. Ahora vamos a estudiar otro tipo de bucles.

EL BUCLE FOR ... END FOR

Hay una segunda forma de hacer que la máquina repita un conjunto de acciones un determinado número de veces. Y decimos *número* con propiedad, pues la condición que va a hacer que el bucle se interrumpa no es que una cadena literal sea igual a otra, sino que un contador alcance cierto valor. Este tipo de bucle es FOR ... NEXT en otras máquinas, pero en el QL también puede ser FOR ... END FOR. Las instrucciones que intervienen son FOR y NEXT, o bien FOR y END FOR. Las acciones que se han de repetir se incluyen entre FOR y NEXT (o entre FOR y END FOR). El QL permite utilizar FOR de dos formas distintas. En la primera no es necesario poner NEXT ni END FOR, a condición de que todas las acciones del bucle estén incluidas en una sola línea de programa. La figura 4.5 es un ejemplo de este método. La línea que contiene el FOR tiene que incluir también el nombre

```
10 CLS
20 FOR N=1 TO 10:PRINT"Reglas basicas del QL"
```

Fig. 4.5 Un bucle FOR sencillo que no requiere NEXT ni END FOR.

de la variable que se usa como contador y los números límite de esa variable, es decir, los valores que debe tener al principio y al final de la ejecución. En el ejemplo, la variable es N y sus valores límite son 1 y 10. La acción que realiza el bucle es sencillamente una instrucción PRINT. Puesto que FOR y la acción correspondiente están en la misma línea, no es necesario señalar el fin del bucle con NEXT ni END FOR. El efecto del programa es escribir en la pantalla diez veces «Reglas básicas del QL». En la primera pasada por el bucle, el valor de N es 1 y se escribe la frase por primera vez. Al llegar al final de la línea, el ordenador incrementa el valor de N, en este caso de 1 a 2. A continuación comprueba si el nuevo valor de N excede de 10. Como no es así, vuelve a ejecutar la instrucción PRINT y a incrementar N, y así sucesivamente hasta que el valor actualizado de N sea mayor que 10; más adelante volveremos sobre esta cuestión. El efecto neto de este programa es realizar la acción prescrita 10 veces.

Pero este bucle es demasiado sencillo. El QL permite construir bucles con varias líneas de instrucciones entre el principio y el fin del bucle. Antes de empezar a complicar las cosas, veamos cómo podríamos haber escrito el programa anterior en una forma más general (Fig. 4.6). En este ejemplo la instrucción FOR N = 1 TO 10 está sola en una línea; a continuación viene PRINT, y la línea 40 señala el final del bucle. END FOR tiene que ir seguido

```
10 CLS
20 FOR N=1 TO 10
30 PRINT"Reglas basicas del QL"
40 END FOR N
```

Fig. 4.6 Bucle FOR ... END FOR típico.

del nombre de la variable que se ha utilizado como contador; de lo contrario, el ordenador emite el mensaje de error «bad line». Pruebe el programa de esta forma y luego modifique la línea 40 (en el apéndice A se explican los métodos de edición) para poner NEXT N en lugar de END FOR N. Comprobará que el programa funciona exactamente igual.

¿De qué nos sirve entonces tener dos formas de interrumpir el bucle? La respuesta está en la mayor flexibilidad que esto permite. Supongamos, por ejemplo, que queremos que el bucle contenga un mensaje alternativo o que pueda ser interrumpido por una condición distinta de la del contador. El BASIC del QL permite programar estas dos posibilidades, como demuestra

```
10 CLS
20 FOR N=1 TO 10
30 J=RND(1,10)
40 IF J=5 THEN EXIT N
50 PRINT"Reglas basicas del QL"
60 NEXT N
70 PRINT"Esta vez hemos llegado a 10"
80 END FOR N
90 PRINT"Ya basta"
```

Fig. 4.7 Comprobación que permite interrumpir un bucle antes de que la variable del contador haya alcanzado su valor máximo.

el programa de la figura 4.7. El bucle es del tipo que ya conocemos, con un contador de 1 a 10. En la línea 30 se genera un número aleatorio comprendido entre 1 y 10 mediante la función RND(1,10); la línea 40 utiliza la condición J = 5 para interrumpir el bucle. Otras máquinas se vuelven locas si se interrumpe un bucle de esta manera, pero el QL controla la situación sin problemas. Si se cumple la condición (IF J = 5), la instrucción EXIT, seguida del nombre de la variable, interrumpe el bucle. Si no se cumple, la acción del bucle continúa con la línea 50, hasta llegar a la 60, donde está NEXT. Así pues, el bucle puede terminar antes de que el contador llegue a 10, y en

tal caso el programa no llegará a la línea 70. Esta línea sólo se puede alcanzar si se han ejecutado las diez pasadas por el bucle, ya que está *antes* de la instrucción END FOR N. En este caso END FOR N marca el final absoluto de todas las acciones del bucle. En otras versiones de BASIC, este tipo de interrupción de bucles es bastante complicado de programar; con el BASIC del QL resulta muy sencillo.

```
10 CLS
20 FOR N=1 TO 10
30 PRINT"El contador marca ";N
40 FOR J=1 TO 500
50 NEXT J:CLS
60 END FOR N
70 PRINT"FIN"
```

Fig. 4.8 Programa con bucles anidados, uno dentro del otro.

Los bucles no tienen por qué ser únicos. En la figura 4.8 se da un ejemplo de *bucles anidados*, es decir, de un bucle dentro de otro. En casos como éste a los bucles les llamamos *interno* y *externo*. El bucle externo empieza en la línea 20 y utiliza la variable N, con valores extremos 1 y 10. La línea 30 forma parte del bucle externo; escribe el valor del contador N. Las líneas 40 y 50 contienen un bucle distinto e independiente. En él hay que utilizar como contador otra variable, y tiene que empezar y terminar antes del final del bucle externo. Para el contador de este bucle interno hemos elegido la variable J; entre FOR y NEXT no hemos intercalado instrucciones, de forma que lo único que hace el bucle es perder tiempo para permitir observar las acciones del bucle externo. La última instrucción del bucle externo es borrar la pantalla (línea 50). El efecto global del programa es visualizar en la pantalla un contador, suficientemente lento como para ser observable. En este ejemplo hemos utilizado END FOR para señalar el final del bucle principal. Tanto con END FOR como con NEXT, es necesario especificar el nombre correcto de la variable del contador al finalizar cada bucle. Si se pone el nombre de una variable (por ejemplo, K) no definida previamente, el ordenador emite el mensaje de error «At line 50 not found» (en la línea 50, no lo encuentro). Si se intercambian las variables, poniendo NEXT N en la línea 50, el contador resultará bastante rápido.

Ya en este momento se puede apreciar lo útil que puede ser un bucle FOR . . . END FOR, pero aún tenemos que sacarle más partido. Para empezar, los bucles que hemos considerado hasta ahora contaban todos hacia arriba, incrementando la variable numérica. No siempre es esto lo que nece-

sitamos. Para modificar la forma en que se actualiza el contador podemos añadir la palabra STEP (paso) al final de la instrucción FOR. Una instrucción completa podría ser, por ejemplo,

FOR N = 1 TO 9 STEP 2

La sucesión de valores de N sería 1, 3, 5, 7 y 9. Si omitimos STEP, el contador se incrementa de 1 en 1.

La figura 4.9 muestra un programa en el que el bucle principal tiene paso −1 y que, por consiguiente, cuenta hacia atrás. N tiene valor inicial 10 y el programa lo decrementa con cada pasada del bucle. La línea 40 es un retardo

```
10 CLS
20 FOR N=10 TO 1 STEP -1
30 PRINT"Quedan ";N;" segundos"
40 FOR j=1 to 500:NEXT j
50 CLS
60 END FOR N
70 PRINT"DESPEGUE!"
```

Fig. 4.9 Programa que cuenta hacia atrás.

que hace que la cuenta atrás tenga lugar a velocidad más «humana». En esta línea no se puede omitir el NEXT j, pues, de hacerlo así, el retardo no funciona. Esta es una forma de retardar el contador. En cambio, no hay forma sencilla de acelerarlo. Muchos ordenadores permiten el uso de variables enteras como contador, de forma que los bucles pueden empezar con

FOR N% = 1 TO 10

pero el QL rechaza esta instrucción con el mensaje «bad line». Es lamentable, porque el QL no se distingue precisamente por la velocidad a la que ejecuta los bucles.

En ocasiones tenemos que utilizar el valor del contador una vez terminado el bucle. Es necesario saber con seguridad cuál será ese valor; la figura 4.10 nos da la solución. Este programa contiene dos bucles, uno que cuenta hacia arriba y el otro hacia abajo. Al final de cada bucle se escribe el valor de la variable utilizada como contador. Así, vemos que N es 5 en la línea 50, después de completar el bucle FOR N = 1 to 5, y que es 1 en la línea 90, cuando se ha completado el bucle FOR N = 5 TO 1 STEP −1. Cuando se va a aprovechar la variable del contador, se debe tener en cuenta que su valor es el especificado en la instrucción de inicio del bucle. Esto es una peculiaridad del QL; en la mayor parte de los ordenadores la variable del contador se mo-

```
10 CLS
20 FOR N=1 TO 5
30 PRINT N
40 NEXT N
50 PRINT"N es ahora ";N
60 FOR N=5 TO 1 STEP -1
70 PRINT N
80 NEXT N
90 PRINT"N es ahora ";N
```

Fig. 4.10 Determinación del valor del contador cuando ha terminado el bucle.

difica una vez más que las especificadas por la instrucción FOR. Si el lector piensa adaptar programas escritos para otras máquinas, no olvide tener esto en cuenta.

Una de las características más interesantes de los bucles FOR ... END FOR consiste en que en su definición se pueden utilizar variables numéricas en lugar de números. En el ejemplo de la figura 4.11 a las variables A, B

```
10 CLS
20 A=2:B=5:C=10
30 FOR N=A TO B STEP B/C
40 PRINT N
50 NEXT N
```

Fig. 4.11 Bucle en el que los valores límite y el paso son variables numéricas.

y C se les asignan valores en la forma habitual (línea 20), y luego se las incluye en la instrucción FOR (línea 30). Los límites son A y B; el paso es la expresión B/C. La regla es que en la instrucción FOR es válido todo lo que sea o represente un número, en sí mismo o como resultado de un cálculo.

BUCLES Y DECISIONES

Ha llegado el momento de utilizar los bucles, y no simplemente demostrar cómo funcionan. Una aplicación sencilla consiste en totalizar números. Queremos escribir un programa que capte datos numéricos por el teclado y haga que el ordenador mantenga y visualice su suma total. Por lo que hemos aprendido hasta ahora, la solución es inmediata si sabemos de antemano cuántos números vamos a sumar. Pero el problema es que si programamos

un bucle del tipo FOR ... NEXT o FOR ... END FOR, tenemos que poner en la instrucción FOR el número total de datos que vamos a sumar. Preferible sería disponer de un programa que se detuviera cuando le hiciéramos una señal convenida, por ejemplo, introducir un número tal como el 0 o el 999. Un número de este tipo se denomina *finalizador*; tiene que ser algo evidentemente distinto de los datos «normales». Para un programa que sume números, es adecuado utilizar como finalizador el 0, porque no importa que el programa lo sume.

```
10 CLS:PRINT"PROGRAMA TOTALIZADOR"
20 AT 1,6:PRINT"Introduzca los numeros uno a uno
"\"segun se los vaya pidiendo"
30 total=0
40 REPeat entrada
50 INPUT"Por favor, un numero ";N
60 IF N=0 THEN EXIT entrada
70 total=total+N
80 PRINT"La suma actual es ";total
90 END REPeat entrada
100 PRINT"La suma final es ";total
```

Fig. 4.12 Programa totalizador que capta y suma números hasta que se teclea el 0.

Veamos el ejemplo de la figura 4.12. En este caso no podíamos utilizar un bucle FOR ... END FOR porque no sabemos cuántos números se van a introducir. Así pues, hemos programado una rutina del tipo REPEAT. Las primeras líneas escriben en la pantalla instrucciones para el usuario. En la línea 30 inicializamos la variable «total». Cuando se ejecuta un programa con la orden RUN, el ordenador inicializa a cero todas las variables automáticamente; en cambio, si se ejecuta el programa de otra forma, por ejemplo, con GOTO 10, las variables conservan el valor anterior. El lector puede pensar que él nunca haría una cosa así, pero eso lo dicen todos. Acostúmbrese, pues, a inicializar al principio de cada programa todas las variables que intervengan en él. La línea 40 es el comienzo de la rutina REPEAT; cada vez que se introduce un número en respuesta a la línea 50, éste es asignado a la variable N. La línea 60 comprueba el valor actual de N. Si es cero, actúa la instrucción EXIT y se ejecuta la línea siguiente a la que contiene END REPEAT. Si el valor no es cero, la línea 70 lo suma a total y la línea 80 escribe la suma actual. La línea 90 es el final de la rutina, END REPEAT.

El efeco global es, pues, el siguiente: si el número captado por la línea 50 no es cero, la línea 90 reconduce el programa a la 50. Esto continúa hasta

que se introduce un cero. Cuando el número introducido es cero, la condición de la línea 60 se satisface y el bucle se interrumpe; el programa continúa en la instrucción inmediatamente siguiente a END REPEAT, es decir, en la línea 100, la cual escribe el valor final de la suma y termina el programa. Haga ahora la siguiente prueba: ejecute el programa; introduzca la primera vez un número, digamos el 2, pero la segunda vez pulse ENTER sin escribir ningún número. El ordenador emite el mensaje «At line 50 error in expresion» (error en una expresión en la línea 50). El mismo error se produce si el dato introducido es una letra. Hay que tener cuidado con los programas totalizadores porque si se pulsa ENTER antes de teclear un número el programa se detiene. Podemos evitarlo captando una cadena literal N$ en lugar de N y comprobándola antes de utilizarla. Todo se andará.

Un bucle de este tipo es mucho más versátil que un FOR ... END FOR, pues no obliga a saber de antemano cuántas veces va a ser ejecutado. La clave de su funcionamiento está en la instrucción IF, y de ella vamos a seguir hablando.

DECISIONES

Podemos realizar diversas comparaciones entre números y variables numéricas (véase la figura 4.13). El ordenador utiliza por comodidad los signos matemáticos; el lector tendrá que recordar cuál es el signo de «mayor que» y cuál el de «menor que». Es importante observar que el signo = significa

Signo	Significado
=	Identidad.
= =	Igualdad aproximada (diferencia menor que una parte en 10 millones).
>	El número que está a la izquierda es mayor que el de la derecha.
<	El número que está a la izquierda es menor que el de la derecha.

Estos signos se pueden combinar de las siguientes formas:

< >	Números distintos
> =	Número de la izquierda mayor o igual que el de la derecha.
< =	Número de la izquierda menor o igual que el de la derecha.

Nota: cuando se combinan los signos < y > con =, éste se tiene que escribir en último lugar. Una combinación tal como = > da lugar al mensaje de error «bad line».

Fig. 4.13 Signos matemáticos que se utilizan para comparar números y variables numéricas.

«idéntico a», de modo que si, por ejemplo, A es 3.9999999 y B es 4.0000000, la condición A = B es falsa porque A y B no son idénticos. El problema está en que el ordenador redondea los números antes de escribirlos en la pantalla. Por ejemplo, en este caso PRINT A podría dar como resultado un 4, y sin embargo la comprobación se realiza antes de redondear los números. Si en lugar del signo = se pone = =, que significa «aproximadamente igual a», este problema queda resuelto. La condición A= =B es verdadera si la diferencia entre A y B es menor que una parte en diez millones. Ésta es una característica exclusiva del QL.

```
10 CLS
20 PRINT"Teclee S o N"
30 PRINT"y luego pulse ENTER"
40 REPeat pregunta
50 INPUT A$
60 IF A$="S" THEN
70 PRINT"La respuesta es SI"
80 EXIT pregunta
90 ELSE IF A$="N"
100 PRINT"La respuesta es NO"
110 EXIT pregunta
120 ELSE PRINT"Por favor, teclee solamente S o N
"
130 END REPeat pregunta
```

Fig. 4.14 Comprobación de variables literales; en este ejemplo, para determinar si una respuesta ha sido S o N.

La figura 4.14 muestra otro tipo de comparación, esta vez con variables literales. Las líneas 20 y 30 escriben instrucciones para el usuario; éste debe teclear una S o una N. En la línea 40 empieza el bucle que capta esa respuesta con la instrucción INPUT de la línea 50. El usuario debe pulsar una tecla y luego ENTER; cualquiera que sea la respuesta, es asignada a la A$, de modo que A$ tendría que ser «S» o «N». Las líneas 60 a 120 analizan esta cadena. Si la tecla pulsada fue la S, la línea 70 escribe «La respuesta es SI» y entra en acción la instrucción EXIT de la línea 80. Si A$ es «N», se escribe el mensaje de la línea 100 y entra en acción el EXIT de la línea 110. Si la tecla pulsada no era ni S ni N, el programa salta a la línea 120, en la cual se recuerda pacientemente al usuario cuáles son las respuestas aceptables. Como la instrucción siguiente es END REPEAT, la ejecución continúa en la línea 40.

En este ejemplo la comprobación consiste en determinar si la respuesta era idéntica a la esperada. La frase «La respuesta es SI» se escribe sólo en el caso de que A$ sea idéntica a «S». Si el usuario teclea un espacio antes o después de la S, o la teclea en minúscula, A$ no será idéntica a «S» y la condición de la línea 60 no se cumplirá. Para prever estas posibilidades habrá que modificar el programa adecuadamente. El QL es uno de los pocos ordenadores que permiten generalizar la instrucción IF ... THEN con ELSE, lo cual constituye otro ejemplo de cómo el QL facilita la programación, a diferencia de otros ordenadores dotados de lo que podríamos llamar «BASIC de la Edad Media».

La línea 120 es una «trampa para errores». En muchos programas al usuario se le ofrece una posibilidad de elección entre varias alternativas; el programador tiene que estar seguro de que sólo se acepten las respuestas que cumplan ciertas condiciones. Una trampa para errores es una sección de un programa que actúa cuando el usuario teclea una respuesta incorrecta. Una buena trampa debe indicar al usuario en qué ha consistido el error y recordarle cuáles son las respuestas correctas. La implementación de una trampa para errores es esencial si se quiere evitar que el programa haga lo que no deba o que se detenga con un mensaje de error. Para un usuario experimentado (y el lector lo será cuando termine de leer este libro), tal accidente no tendría demasiada importancia, pero un usuario no experto podría no saber qué hacer ante un mensaje de error. Un buen programa tiene que evitar todos los posibles errores, pues si el programa se detiene puede ocurrir que se pierda toda la información que el usuario ha introducido hasta ese momento. El lector ya sabe cómo evitar el desastre: teclear GOTO seguido del número de línea correcto y pulsar ENTER, suponiendo, desde luego, que se sepa en qué línea continuar. Pero si el usuario no ha escrito el programa y tiene un listado a mano, la pérdida puede consumarse. Las trampas de errores son el medio de evitar que un error detenga el programa.

Para dar un ejemplo más de la potencia de estas sencillas instrucciones, veamos el programa de la figura 4.15: un juego en el que se trata de acertar un número. En la línea 10 empieza la rutina que se ejecuta cuando el usuario no acierta el número; la línea 20 borra la pantalla; la instrucción RND (1,10) asigna a X un número entero comprendido entre 1 y 10. No podemos predecir cuál va a ser ese número porque, como el lector sabe, RND elige un número al azar. La elección es suficientemente aleatoria para juegos, pero no para aplicaciones estadísticas. En las líneas 30 y 40 se pide al usuario que trate de adivinar el número, exacta o aproximadamene. La línea 50 capta la respuesta; las líneas 60, 90 y 120 la examinan. Si la respuesta es idéntica al número aleatorio, se escribe el mensaje de la línea 70 y el EXIT de la línea 80 hace que el programa termine. Una comprobación no tan obvia es la de la línea 90. La expresión N − X es la diferencia entre el número tecleado por

```
10 REPeat adiv_num
20 CLS:X=RND(1,10)
30 AT 10,2:PRINT"Adivine el numero!"
40 AT 2,4:PRINT"Si se aproxima, ya se lo dire"
50 INPUT N
60 IF N=X THEN
70 PRINT"ACERTO!"
80 EXIT adiv_num
90 ELSE IF ABS(N-X)<3 THEN
100 PRINT"Caliente. El numero era ";X
110 EXIT adiv_num
120 ELSE IF N<>X THEN
130 PRINT"Frio. Adivine otra vez"
140 FOR J=1 TO 1000:NEXT J
150 END IF
160 END REP adiv_num
170 PRINT"FIN DEL JUEGO"
```

Fig. 4.15 Juego de adivinación de números basado en comparaciones entre números.

el usuario, N, y el número X. Dependiendo de cuál sea la respuesta, N – X puede ser negativo o positivo. En cualquier caso, ABS convierte N – X en positivo. Si la diferencia es 1 o 2, es decir, menor que 3, se escribe el mensaje de la línea 100. Si la diferencia es mayor, el programa se repite a causa del ELSE de la línea 120. La comprobación de esta línea parece innecesaria, ya que si el programa llega a ella N no puede ser igual a X. La razón está en que, para que ELSE funcione, después de esta palabra hay que poner alguna comparación. El efecto es escribir el mensaje y hacer que el proceso se repita después de pasar por 160. El lector ya habrá encontrado varios defectos en este programa. ¿Por qué terminar el programa cuando la respuesta ha sido acertada? ¿Por qué no continuar hasta que el usuario teclee un 0? ¿Por qué no programar un tanteador? Se podría dar 3 puntos por cada respuesta acertada y 1 por cada aproximación. Intente incluir alguna de estas modificaciones, pues la única forma de aprender a programar es experimentar con los programas.

RESPUESTA CON UNA SOLA TECLA

Hasta ahora nuestros programas han captado las respuestas de tipo S/N mediante instrucciones INPUT; esto requiere teclear la respuesta y luego pulsar

ENTER. La ventaja de este método es que permite rectificar la respuesta, siempre que la rectificación se haga antes de pulsar ENTER. Para mayor agilidad, se puede utilizar la función INKEY$. Esta instrucción realiza una comprobación del teclado para determinar si se está pulsando una tecla. La comprobación es muy rápida, y la única forma de utilizarla es incluirla en

```
10 CLS
20 PRINT"Pulse cualquier tecla ..."
30 REPeat prueba
40 K$=INKEY$:IF K$<>"" THEN
50 EXIT prueba
55 END IF
60 END REPeat prueba
70 PRINT"La tecla ha sido ";K$
```

Fig. 4.16 Detección de la pulsación de una tecla con INKEY$. Algunas teclas hacen que el programa funcione y sin embargo no envían ningún carácter a la pantalla.

KEYROW ()

↓								
7	SHIFT	CTRL	ALT	X	V	/	N	,
6	8	2	6	Q	E	O	T	U
5	9	W	I	TAB	R	–	Y	
4	L	3	H	1	A	P	D	J
3	1	CAPS LOCK	K	S	F	=	G	;
2	1	Z	·	C	B	£	M	~
1	ENTER	←	↑	ESC	→	\	SPACE	↓
0	F4	F1	5	F2	F3	F5	4	7
CODIGOS →	1	2	4	8	16	32	64	128

Fig. 4.17 Función KEYROW. Para cada fila, cada tecla genera un código que es detectado por KEYROW.

un bucle que se repita hasta que se detecte la pulsación. El programa de la figura 4.16 es un ejemplo de este método. La instrucción INKEY$ produce una cadena literal cuando se pulsa una tecla, de forma que podemos asignar INKEY$ a una variable literal, K$. Así, cuando se pulse una tecla, el carácter generado por ella será asignado a K$; mientras K$ sea una *cadena vacía*, el bucle seguirá repitiéndose. Observe el lector que la cadena vacía se escribe con dos signos de comillas sin nada en medio (ni siquiera espacio). Si ejecuta el programa de la figura 4.16 puede observar el efecto de pulsar las diferentes teclas. Hay algunas teclas, como por ejemplo la barra espaciadora, que no producen ningún signo visible en la pantalla (línea 70), y sin embargo hacen que se interrumpa el bucle (línea 50). Otras teclas, como las de función de la izquierda del teclado, producen un signo en forma de rombo, y otras, finalmente, que no tienen ningún efecto.

Hay otro método diferente de determinar si se está pulsando una tecla. La instrucción en que se basa es KEYROW. El teclado está dividido, a estos efectos, en ocho grupos, numerados del 0 al 7. Cada grupo contiene 8 teclas, y cada tecla genera un código. En la figura 4.17 se ilustran los números de fila y los códigos generados por las teclas. Por ejemplo, para la fila número 3, la tecla «S» da el número 8; ninguna otra tecla produce el mismo código *en esta fila*. KEYROW permite examinar teclas específicas, sin que interfieran las demás.

En la figura 4.18 se muestra cómo utilizar la instrucción KEYROW. Esta palabra debe ir seguida de un número (entre paréntesis), que es el número de fila, del 0 al 7. KEYROW(X) es una función cuyo valor se puede escribir en la pantalla, o asignar a una variable o comprobar en una instrucción IF. En este ejemplo hemos establecido un bucle que comprueba continuamente el valor de KEYROW(1). Si este valor es 64, el bucle se interrumpe porque se ha pulsado la barra espaciadora. Si el valor es distinto, el bucle continúa indefinidamente. KEYROW es una forma particularmente útil de examinar teclas, especialmente en la programación de juegos, ya que permite comprobar todas las teclas, incluso las de movimiento del cursor y las de función.

```
10 CLS
20 PRINT"PULSE LA BARRA ESPACIADORA"\"PARA CONTI
NUAR"
30 REPeat bucle
40 IF KEYROW(1)=64 THEN
50 EXIT bucle
60 END IF
70 REN REPeat bucle
80 PRINT"ESO ES UN ESPACIO!"
```

Fig. 4.18 Detección de una tecla concreta con KEYROW; en este ejemplo, la barra espaciadora.

5

Cadenas literales

En el capítulo 3 hemos estudiado, aunque brevemente, unas cuantas funciones numéricas. Si al lector le gustan los números, tanto mejor. Pero las funciones literales pueden ser bastante más interesantes. Lo que las hace tan atractivas es el hecho de que las acciones más espectaculares de que es capaz el ordenador las realiza frecuentemente por medio de *funciones literales.* Empecemos por decir qué es una función literal. Para nuestros propósitos, función literal es toda acción que puede ser efectuada con cadenas literales. Por si esta definición no deja las cosas claras, entremos en detalle.

Para el QL una cadena literal es un conjunto cualquiera de caracteres. Las cadenas pueden ser representadas por *variables literales*, cuyo nombre, como ya sabemos, tiene que terminar en un signo de dólar ($). En el QL una cadena puede contener hasta 32768 caracteres, mucho más de lo que normalmente se puede necesitar; en otros ordenadores la longitud máxima de las cadenas es 255 caracteres. Otra particularidad del QL es que si una cadena consta de solamente dígitos, se la puede manejar como si fuera un número. En lo que sí coincide el QL con otras máquinas es en la forma de almacenar las cadenas, utilizando lo que llamamos código ASCII. Estas letras son las siglas de «American Standard Code for Information Interchange» (código americano estándar para el intercambio de información). El código ASCII es el utilizado por la mayor parte de los ordenadores. En la figura 5.1 se muestra un listado de los códigos ASCII junto con los caracteres correspondientes.

Cada carácter está representado por un número, del 32 (que es el espacio) al 127. En mi impresora el carácter 127 es un cuadrado negro, pero en la pantalla del QL es un signo parecido a la letra griega sigma. Se pueden asignar caracteres a una variable literal utilizando el signo =, con o sin LET. Al hacer la asignación de esta manera, los caracteres tienen que estar encerrados entre comillas. Cuando la asignación se realiza por medio de la ins-

32		33	!	34	"	35	#	36	$
37	%	38	&	39	'	40	(	41	)
42	*	43	+	44	,	45	-	46	.
47	/	48	0	49	1	50	2	51	3
52	4	53	5	54	6	55	7	56	8
57	9	58	:	59	;	60	<	61	=
62	>	63	?	64	@	65	A	66	B
67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L
77	M	78	N	79	O	80	P	81	Q
82	R	83	S	84	T	85	U	86	V
87	W	88	X	89	Y	90	Z	91	[
92	\	93	]	94	^	95	_	96	`
97	a	98	b	99	c	100	d	101	e
102	f	103	g	104	h	105	i	106	j
107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t
117	u	118	v	119	w	120	x	121	y
122	z	123	{	124	|	125	}	126	~
127	■								

Fig. 5.1 Caracteres del código ASCII.

trucción INPUT, las comillas no son necesarias; pero sí lo son, en cambio, cuando las cadenas están contenidas en líneas DATA para ser leídas por instrucciones READ.

La codificación de las variables numéricas es muy diferente. Para representar una variable numérica el ordenador utiliza siempre el mismo número de códigos, tanto si el número es grande como si es pequeño. Puesto que una cadena literal consiste en una sucesión de códigos almacenados en la memoria del ordenador, podemos hacer con las cadenas manipulaciones que no son posibles con los números. Por ejemplo, podemos determinar muy fácil-

mente cuántos caracteres tiene una cadena. Podemos extraer determinados caracteres de ella, o también cambiar alguno o insertar otros nuevos. Las acciones de este tipo son lo que denominamos funciones literales.

LEN

Hemos mencionado que una de las funciones literales posibles es determinar el número de caracteres de que consta una cadena. Puesto que las cadenas pueden llegar a tener hasta 32767 caracteres, es muy conveniente disponer de un método automático para contarlos. El método lo proporciona la función LEN. La palabra LEN debe ir seguida del nombre de la variable literal, entre paréntesis. El resultado es siempre un número, que en consecuencia podemos manejar como tal; por ejemplo, escribiéndolo en la pantalla o asignándolo a una variable numerica. No es necesario dejar espacio entre la N de LEN y el signo de abrir paréntesis.

```
10 CLS
20 A$="EL FABULOSO QL"
30 PRINT"Hay ";LEN(A$);" caracteres en la frase
";A$
40 INPUT"Teclee usted otra:"\k$
50 PRINT k$\" tiene ";LEN(k$);" caracteres"
```

Fig. 5.2 Presentación de LEN, primer miembro de la familia de funciones literales.

La figura 5.2 muestra un ejemplo de utilización de LEN. La línea 20 asigna una cadena a una variable literal; la línea 30 escribe el número de caracteres de que consta la cadena (es decir, su longitud). Observe que decimos «caracteres», no «letras». Tanto los diversos signos como el espacio cuentan como caracteres, ya que todos están representados por un código ASCII. Las líneas 40 y 50 calculan y escriben la longitud de la cadena tecleada por el usuario. Esto puede ser útil para asegurarse de que un nombre introducido por el teclado no es excesivamente largo para una aplicación concreta. Por ejemplo, podría tratarse de un programa que escribiera nombres en un formulario. Si se escribiera un nombre demasiado largo, parte de él aparecería en la columna correcta, y el resto en la siguiente. Quizá haya observado, si su nombre o dirección son suficientemente largos, que cuando recibe cartas con etiqueta impresa por ordenador parte del nombre o de las señas está abreviado. Ahora ya sabe por qué.

Esto no parece demasiado impresionante, pero podemos sacarle prove-

```
10 CLS
20 T$="El QL en acción"
30 col=(37-LEN(T$))/2
40 AT col,2:PRINT T$
```

Fig. 5.3 Centrado de títulos mediante LEN.

cho, como demuestra el programa de la figura 5.3. En este programa se utiliza LEN como parte de una rutina que escribe una cadena llamada T$ centrada en una línea. Esta rutina es verdaderamente útil, y el lector tendrá ocasión de utilizarla repetidamente en sus propios programas porque le ahorrará el tedioso trabajo de contar los caracteres del texto que quiera escribir centrado. LEN calcula la longitud de la cadena T$. Este número se resta de 37, y el resultado se divide por 2. Si la longitud de la cadena es par, el número «col» contendrá un 0.5, pero esta parte decimal es ignorada por AT en la línea 40. Dicho sea de paso, se puede utilizar el número 37 o el 38. En cualquier caso, los textos quedan razonablemente bien centrados. El 38 funciona mejor cuando la longitud es par; el 37 cuando la longitud es impar. Como el lector ya estará pensando, se puede afinar más y elegir 37 o 38, según convenga, utilizando algún IF ... THEN ... ELSE.

La rutina entera se puede programar en una sola línea. Aquí la hemos escrito en tres líneas para mayor claridad. Una rutina así programada se puede aplicar a cualquier cadena cuyo nombre sea T$. En el próximo capítulo estudiaremos los *procedimientos*; son zonas de un programa que se teclean una sola vez y pueden ser utilizadas por el programa repetidamente.

DISECCIÓN DE CADENAS

El siguiente grupo de funciones literales está formado por lo que denominamos *operaciones de disección* o de *corte*. El resultado de cortar una cadena es otra cadena: un trozo tomado de la cadena original. La disección permite averiguar qué caracteres están presentes en diversos lugares de una cadena.

Esto parece irresistiblemente interesante, así que vamos a ver enseguida un ejemplo, figura 5.4. En la línea 20 se asigna una cadena a la variable a$, y en la línea 30 otra cadena a b$. En la línea 40 se escriben las dos cadenas, separadas por un espacio. Hasta aquí todo es conocido. En la línea 60 se define una nueva cadena; la línea 70 la escribe, «Q.L.». ¿Como ha ocurrido esto? La clave está en a$(1) y b$(1). El número 1 hace referencia a la posición de los caracteres dentro de la cadena, empezando a contar por la izquierda. Por lo tanto, a$(1) es la Q de «QUE»; b$(1) es la L de «LA». Si hubiéramos

```
10 CLS
20 a$="QUE MAQUINA ES"
30 b$="LA MAS AVANZADA?"
40 PRINT a$;" ";b$
50 PRINT:
60 c$=a$(1)&"."&b$(1)&"."
70 PRINT" El ";c$
```

Fig. 5.4 Extracción de iniciales.

puesto 2 en lugar de 1, las letras extraídas habrían sido «U» y «A»; el mensaje ya no tendría sentido.

Copiar una letra de una cadena no es extraordinariamente útil, aunque puede servir, por ejemplo, para extraer las iniciales de un nombre. El proceso de disección es más útil si especificamos dos números en lugar de uno. El primero de ellos indica la posición del primer carácter; el otro, la del último. Veamos el ejemplo de la figura 5.5. Las letras que se extraen son de la 1.ª a la 3.ª, «Tel», utilizando A$(1 TO 3). Obsérvese el espacio que hay que dejar a ambos lados de la palabra TO.

```
10 CLS
20 A$="TELEFONO"
30 PRINT A$(1 TO 3)
```

Fig. 5.5 Extracción de varias letras de una cadena con TO.

DISECCIÓN POR LA DERECHA

La disección de cadenas no se limita a copiar unos cuantos caracteres tomados de su izquierda. También podemos extraer caracteres de la derecha de una cadena. En la figura 5.6 se extrae la última palabra de una frase. Por supuesto, las aplicaciones prácticas son más serias. Por ejemplo, se pueden extraer los cuatro últimos dígitos de una cadena tal como 91-746-4276. Y de-

```
10 CLS
20 A$="QL MAGICO"
30 PRINT"DEMASIADO "& A$(4 TO) & "PARA MI"
```

Fig. 5.6 Extracción de caracteres de la derecha de una cadena.

cimos «cadena» deliberadamente, porque un dato como éste tiene que ser tratado como cadena, no como número. En efecto, si tratamos de asignar este dato a una variable numérica, el ordenador no entenderá lo que realmente queremos hacer; es decir, si escribimos N = 91-746-4276, el ordenador supondrá que queremos restar 746 de 91, y 4276 del resultado. Evitemos problemas haciendo N$ = "91-746-4276". Observe que en el ejemplo de la figura 5.6 no es necesario especificar la posición del último carácter de la cadena. Desde luego, podríamos calcular esa posición con LEN, pero no hace falta. Cuando no se pone ningún número a continuación de TO, el ordenador supone que lo que se desea extraer es el resto de los caracteres hasta el final de la cadena.

```
10 CLS
20 INPUT"Por favor, escriba su nombre"\nombre$
30 N=LEN(nombre$)
40 FOR J=1 TO N
50 PRINT nombre$(1 TO J);"  ";nombre$(J TO)
60 NEXT J
```

Fig. 5.7 Disección por ambos lados de una cadena.

Podemos conseguir efectos llamativos utilizando estas instrucciones de disección. El programa de la figura 5.7 hará diabluras con las letras de su nombre. En la línea 20 se capta el nombre y se lo asigna a la variable nombre$, incluidos espacios y guiones (esperemos que nadie necesite asteriscos ni barras inclinadas). En la línea 40 empieza un bucle cuyo límite superior es el número de caracteres. La línea 50 es la fundamental. Cuando J es 1, esta línea escribe la primera letra de su nombre a la izquierda de la pantalla, y el resto del nombre dos lugares más a la derecha. En la siguiente pasada del bucle se escriben los dos primeros caracteres, luego dos espacios y después el resto del nombre. Así se continúa hasta escribir el nombre completo a la izquierda.

DISECCIÓN POR EL CENTRO

Hay una variante de la disección de cadenas que permite extraer un tramo cualquiera de una cadena dada. La palabra que la realiza sigue siendo TO, acompañada de dos parámetros. El número que precede a TO es la posición de la primera letra que se debe extraer; el número que sigue a TO es la posición de la última letra.

```
10 CLS
20 A$="Referencia N. PDQ123 #7146"
30 PRINT"La referencia para las tiendas es ";A$(
15 TO 20)
```

Fig. 5.8 Extracción de un tramo cualquiera de una cadena. Esta instrucción puede ser controlada por variables.

Para ver esta instrucción en acción consideremos el programa de la figura 5.8. La línea 20 asigna a A$ una cadena; la línea 30 escribe una frase tomando caracteres de la cadena original.

Además de extraer caracteres de una cadena podemos, como dijimos antes, modificar ciertos caracteres en ella. En el programa de la figura 5.9 la línea 10 asigna a una variable literal la palabra PROGRAMA. La línea 20

```
10 A$="PROGRAMA"
20 PRINT A$
30 A$(4 TO 6)="BLE"
40 PRINT A$
```

Fig. 5.9 Modificación de algunos caracteres de una cadena.

la escribe, y la 30 modifica tres de sus letras. Esto no altera la longitud de la cadena, sino que sencillamente cambia GRA por BLE. La línea 40 demuestra que el cambio ha sido realizado. ¿Qué ocurre si intentamos modificar demasiadas letras? Compruébelo el lector haciendo que la línea 30 sea

30 A$(4 TO 9) = "BLEMAS"

y ejecutando otra vez el programa. Como puede observar, la línea 40 sigue escribiendo PROBLEMA. No se puede alargar la cadena por este procedimiento. ¿Podemos acortarla? Teclee otra vez la línea 30:

30 A$(6 TO 8) = "E"

y ejecute el programa. La línea 40 escribe la palabra PROGRE; es decir, la palabra es más corta. En cambio, la cadena sigue teniendo la misma longitud; en efecto, si se teclea PRINT LEN(A$), se comprueba que la longitud de la cadena sigue siendo 8; lo que ocurre es que todos los caracteres posteriores a la E de PROGRE han sido sustituidos por espacios. Esto puede producir confusión si queremos basarnos en la longitud de las cadenas para centrarlas en la pantalla o para escribirlas en columnas.

```
10 CLS
20 INPUT"Por favor, escriba su nombre ";NM$
30 L=LEN(NM$):C=INT(L/2)
40 FOR N=1 TO C
50 AT 19-N,N:PRINT NM$(C-N+1 TO C+N)
60 NEXT N
```

Fig. 5.10 Construcción de una pirámide de letras para ilustrar la función TO combinada con una fórmula.

Una de las características de todas estas instrucciones de disección es que se las puede controlar con variables numéricas en lugar de números. El programa de la figura 5.10 realiza una disección algo más complicada. La línea 20 capta una cadena. La línea 30 realiza unos cálculos; veamos para qué. Supongamos que el nombre tecleado es MANOLO. Esta cadena tiene 6 letras, luego la línea 30 asigna el valor 6 a L; C es la parte entera de L/2, que es 3. En la línea 40 empieza un bucle de tres pasadas. En la primera el ordenador escribe en la columna 18, ya que N = 1 y 19 − N = 18. Lo que escribe es el tramo NM$(3 TO 4) porque, al ser N = 1 y C = 3, resulta que C − N + 1 es 3 y que C + N es 4. En la segunda pasada por el bucle, N es 2, C − N + 1 es 2 y C + N es 5, de modo que se escribe NM$(2 TO 5), o sea, ANOL. El bucle continúa de esta forma; el resultado global es una pirámide de letras tomadas de su nombre. El efecto es más impresionante cuanto más largo sea el nombre.

Advertencia importante

Con la mayor parte de los dialectos de BASIC se pueden utilizar las instrucciones LEFT$, RIGHT$ y MID$. *No lo intente* en el QL. A diferencia de otras órdenes incorrectas, que el QL simplemente ignora, estas instrucciones hacen que mi QL entre en un bucle infinito del que sólo se puede salir pulsando el botón RESET, con la consiguiente pérdida del programa. Es probable que en las versiones posteriores del QL se haya corregido este defecto; en todo caso, téngalo en cuenta cuando convierta programas escritos para otras máquinas.

OTROS CARACTERES

Continuemos estudiando más funciones literales. Si retrocede unas páginas, recordará que hemos hablado del código ASCII, que es el conjunto de nú-

meros con los que se representan los caracteres que podemos escribir en la pantalla. Podemos averiguar qué código corresponde a cada carácter utilizando la función CODE seguida del carácter en cuestión entre comillas y paréntesis, o bien del nombre de una variable literal entre paréntesis (pero sin comillas). El resultado de esta función es un número, el código ASCII correspondiente al carácter examinado. Si se aplica CODE a una cadena, el resultado es el código ASCII del primer carácter. Consideremos el ejemplo de la figura 5.11. La línea 10 asigna una cadena a la variable A$. La línea 30 inicia un bucle que recorre todas las letras de A$. Las letras son extraídas mediante A$(N); la función CODE da el código ASCII de cada una de ellas. Los signos de punto y coma y el espacio de la línea 40 controlan la forma de escritura.

```
10 A$="El ordenador QL"
20 CLS:PRINT
30 FOR N=1 TO LEN(A$)
40 PRINT CODE(A$(N));" ";
50 NEXT N
```

Fig. 5.11 Obtención del código ASCII de las letras mediante la función CODE.

CODE tiene una función inversa, CHR$. El parámetro, entre paréntesis, tiene que ser un código ASCII. Por ejemplo, la instrucción PRINT CHR$(65) escribe en la pantalla la letra A, ya que 65 es el código ASCII de esa letra. Esta función permite la codificación de mensajes. A veces el pro-

```
10 CLS:AT 2,2
20 PRINT"Que significa QL?"
30 AT 2,4:RESTORE
40 PRINT"Pulse una tecla para ver la respuesta"
50 K$=INKEY$:IF K$="" THEN
60 GO TO 50:END IF
70 FOR J=1 TO 12
80 READ N:PRINT CHR$(N);
90 NEXT J
100 STOP
110 DATA 81,117,97,110,116,117,109,32,76,101,97,
112
```

Fig. 5.12 Utilización de los códigos ASCII para ocultar un mensaje en un programa; recuperación del mensaje con CHR$.

gramador desea ocultar un mensaje de forma que no sea evidente para quien lea el listado del programa. Los códigos ASCII no desorientarían a un experto, pero posiblemente sí al usuario inexperto. La figura 5.12 da un ejemplo de esto. Las líneas 50 y 60 contienen un bucle con INKEY$ para esperar por el usuario. Cuando éste pulsa una tecla, el bucle que empieza en la línea 70 escribe 12 letras en la pantalla. Los códigos ASCII de estas letras están en una línea de DATA. La línea 80 los lee, los convierte en caracteres y los escribe (uno a continuación de otro, gracias al punto y coma). Para ocultar mejor los caracteres se podrían utilizar números no tan obvios; por ejemplo, la cuarta parte de los códigos ASCII, o los códigos más 20 multiplicados por 5, etc. Estos nuevos códigos se pueden guardar en la lista DATA y programar su correcta conversión a caracteres ASCII. Esto será suficiente para desanimar a todo curioso, salvo quizá a los más persistentes decodificadores. Dicho sea de paso, en este ejemplo se utiliza READ ... DATA en un bucle; normalmente en las líneas de DATA se guarda sólo la información que el programa utiliza en todas las ejecuciones. Obsérvese la presencia de la instrucción RESTORE (línea 30). Si no la incluyéramos, el programa sólo se podría ejecutar una vez. En la segunda ejecución el ordenador emitiría el mensaje «End of file in 80» (fin de fichero en 80). Esto indica que no tiene más datos que leer, ya que todos los disponibles los ha leído en la primera ejecución. Recuerde el lector que ya hemos hablado de esto en el capítulo 3.

Otra forma de utilizar RESTORE es poner un número de línea a continuación; entonces el primer dato leído será el que figure en primer lugar en la línea especificada. Por supuesto, en esa línea tiene que haber una lista DATA. Veamos el programa de la figura 5.13. La línea 80 ejecuta una instrucción RESTORE que especifica un número de línea que es función del número introducido por el usuario. Por ejemplo, si a es 2, RESTORE 1000*a

```
10 CLS
20 PRINT"Que lista desea?"
30 REPeat elegir
40 INPUT"Elija (1/2/3) ";a
50 SELect a=1 TO 3:EXIT elegir
60 PRINT"Elija solamente de 1 a 3, por favor"
70 END REPeat elegir
80 RESTORE a*1000
90 FOR N=1 TO 4
100 READ X:PRINT X;" ";:NEXT N
110 STOP
1000 DATA 1,2,3,4
2000 DATA 2,4,6,8
3000 DATA 3,6,9,12
```

Fig. 5.13 Utilización de RESTORE para seleccionar una lista de datos.

significa «empezar a leer los datos en la línea 2000». Cada lista DATA contiene cuatro datos, de forma que el bucle de las líneas 90 y 100 lee los cuatro datos de la línea elegida. Ésta es una forma útil de seleccionar una lista cada vez que se ejecuta un programa.

ORDENACIÓN DE CADENAS

En la figura 4.13 vimos cómo se pueden comparar los números. También es posible comparar cadenas literales utilizando códigos ASCII como base para la comparación. Dos letras son iguales si tienen el mismo código ASCII; fácilmente se comprende, pues, qué puede significar el signo = aplicado a cadenas literales. Para que dos cadenas sean iguales, tienen que contener los mismos caracteres en el mismo orden. El significado de < y > ya no es tan evidente. El código ASCII de la letra A es 65, y el de la letra B es 66. Teniendo esto en cuenta, se puede decir que A «es menor que» B porque su código ASCII es menor. Para colocar letras por orden alfabético basta entonces con disponerlas en orden de códigos ASCII crecientes.

Este proceso se puede generalizar para comparar palabras completas letra a letra. El programa de la figura 5.14 compara palabras utilizando los signos = y >. La línea 20 asigna a A$ una «palabra» compuesta por las primeras seis letras de una fila del teclado. La línea 30 pide al usuario que escriba una palabra. Las líneas 40 a 70 realizan las comparaciones. Si la palabra tecleada, asignada a B$, es idéntica a QWERTY, se escribe el mensaje de la línea 50 y el programa conluye. Si QWERTY es posterior en orden alfabético a la palabra introducida, se ejecuta la línea 70. Por ejemplo, si se teclea PERIFERICO, puesto que Q es posterior a P en orden alfabético (y su código

```
10 CLS
20 A$=QWERTY"
30 PRINT:INPUT"Escriba una palabra en mayusculas
 ";B$
40 IF B$=A$ THEN
50 PRINT"Igual que la mia!":STOP
60 ELSE IF A$>B$ THEN
70 Q$=A$:A$=B$:B$=Q$
80 END IF
90 PRINT"El orden correcto es ";A$;" y luego ";B
$
100 STOP
```

Fig. 5.14 Comparación de palabras para ordenarlas alfabéticamente.

ASCII es mayor que el de P), la palabra B$ sería menor que A$, y la línea 70 las intercambia. Esto se realiza asignando A$ a una nueva variable, Q$ (Q$ = "QWERTY"), luego asignado B$ a A$ (A$ = "PERIFERICO") y finalmente Q$ a B$ (B$ = "QWERTY"). La línea 90 escribe primero A$ y luego B$, que siempre estarán en orden alfabético. En efecto, si la palabra tecleada es posterior a QWERTY (por ejemplo, TERMINAL), entonces A$ no es «mayor que» B$ y la comprobación de la línea 60 da como resultado «falso». Las palabras no son intercambiadas y el orden A$ B$ es correcto. Nótese que el programa ordena correctamente palabras tales como QWERTZ y QWERTX; es decir, para el orden no cuenta sólo la primera letra.

LISTAS

Los nombres de variables que hemos manejado hasta ahora han sido útiles, pero su utilidad tiene una seria limitación. Supongamos, por ejemplo, que cierto programa requiere que se introduzca por el teclado una larga serie de datos numéricos. No sería nada práctico tener que asignar una variable a ca-

```
10 CLS:DIM A(10)
20 FOR N=1 TO 10
30 A(N)=RND(10,100)
40 NEXT N
50 PRINT
60 AT 14,2:PRINT"Lista de puntuaciones"
70 PRINT:FOR N=1 TO 10
80 AT 2,N+3:PRINT"El cantante ";N;" ha recibido
";A(N);" puntos"
90 NEXT N
```

Fig. 5.15 Lista de variables numéricas indexadas. La idea es más sencilla de lo que su nombre puede sugerir.

da dato. El programa de la figura 5.15 ofrece una solución a este problema. Las líneas 10 a 40 generan un conjunto de diez puntuaciones. La variable de la línea 30 es nueva para nosotros. Es una *variable indexada*; el *índice* es el número N. Esta terminología no es específica de la informática y viene siendo utilizada desde antes de que los ordenadores existieran. ¿Cuántas veces ha manejado usted una lista de artículos numerados con 1, 2, 3, . . .? Estos números son un índice, que sirve sencillamente para identificar los ele-

mentos. De la misma forma, utilizando las variables A(1), A(2), A(3), ... podemos identificar elementos diferentes que tengan un nombre común, A. Un miembro tal como A(2) tiene un nombre que se lee «A de dos».

La utilidad de este método reside en que permite utilizar un solo nombre para la lista completa y acceder a las variables individuales por su número de índice. Puesto que el índice puede ser también una variable numérica o una expresión, la flexibilidad es extraordinaria. En el programa de la figura 5.15 las líneas 20 a 40 generan los elementos de una lista. Cada elemento se obtiene generando un número aleatorio comprendido entre 1 y 10, y ese número se asigna a A(N). Las líneas 60 a 90 escriben en la pantalla la lista de las puntuaciones.

Pero no hemos explicado una instrucción que aparece por primera vez en este programa: DIM. Si se omite DIM A(10) en la primera línea, el ordenador emite el mensaje «Bad name» (nombre incorrecto) cada vez que se encuentra con A(N). La razón es que el ordenador tiene que ser preparado para manejar este tipo de variables; la preparación consiste en que el ordenador reserve la memoria necesaria para recibir los datos. La instrucción DIM (dimensionar) realiza esa reserva de memoria. En realidad, una instrucción tal como DIM A(10) permite que la lista contenga 11 elementos, porque también podemos utilizar A(0) (pero no A(11)).

En resumen, dimensionar consiste en escribir, después de DIM, todos los nombres de las listas que se vayan a utilizar, poniendo después de cada nombre entre paréntesis el número *máximo* de elementos que vaya a tener la lista.

En el curso del programa no es necesario llegar a alcanzar ese número máximo, pero lo que no se puede hacer es sobrepasarlo. Si esto curriera, el programa se detendría con un mensaje de error; entonces habría que modificar la instrucción DIM y volver a empezar, lo cual no sería nada agradable si se estuviera tecleando una lista de cien números. En una misma instrucción DIM se pueden dimensionar cuantas variables se desee, como muestra la figura 5.16.

En la figura 5.16 se ilustra el manejo de otro tipo de listas. Este programa capta los nombres y puntuaciones de diez artículos. Cuando la lista está completa, se borra la pantalla y se inicializa la variable «Total» (línea 70). A continuación se escribe la lista; en cada pasada por el bucle se actualiza la suma total de puntos (línea 120), dato necesario para escribir la puntuación media al final. Lo que el lector debe observar en este ejemplo es que las listas pueden contener no sólo números, sino también cadenas literales. Un nombre más «matemático» para las listas es el de *vector*; en la figura 5.16 se utilizan vectores literales (nombres) y vectores numéricos (puntuaciones).

Ahora bien, en el QL las listas literales se dimensionan de distinta forma que las listas numéricas. Casi todos los demás ordenadores permiten que am-

```
10 CLS:DIM A(10),N$(10,20)
20 AT 1,10:PRINT"Introduzca nombres y puntuacion
es"
30 FOR N=1 TO 10
40 INPUT"Nombre ";N$(N)
50 INPUT"Puntos ";A(N)
60 NEXT N
70 CLS:Total=0
80 AT 14,2:PRINT"LISTA DE PUNTUACIONES"
90 FOR N=1 TO 10
100 AT 2,N+3:PRINT N$(N);
110 AT 19,N+3:PRINT A(N)
120 Total=Total;A(N)
130 NEXT N
140 PRINT
150 PRINT"La media es ";Total/10
```

Fig. 5.16 Listas numéricas y literales.

bos tipos de listas se dimensionen de la misma manera. El QL, como era de esperar, es diferente. La instrucción DIM, cuando se la aplica a una lista literal, requiere dos parámetros. El primero es el máximo número de elementos que puede contener la lista; el segundo es la longitud máxima de cada uno de esos elementos (cadenas). En este ejemplo el segundo parámetro es 20. Si se teclea un nombre de más de 20 caracteres, el ordenador lo trunca; es decir, se queda solamente con los primeros 20 caracteres e ignora los restantes. Obsérvese que en este ejemplo un nombre de más de 18 letras se solaparía con la columna de las puntuaciones, de forma que esta peculiaridad del QL puede sernos útil. En el QL hay que planificar la utilización de las listas literales más cuidadosamente que en otros ordenadores.

FILAS Y COLUMNAS

Ya conocemos los vectores o listas de elementos. Hay otra variedad de lista, denominada *matriz*. Una matriz es un vector de grupos de elementos, cada uno de los cuales es a su vez un vector. Podemos imaginar una matriz como un conjunto de filas y columnas; cada grupo sería una fila, y los elementos de un grupo determinado estarían en columnas diferentes. Aclaremos estas ideas con el ejemplo de la figura 5.17. En este ejemplo utilizamos la variable N$ con dos índices. El primero es el número de fila; el segundo es el número de columna. Para leer los elementos de esta matriz necesitamos dos bucles

```
10 CLS:DIM N$(3,2,10)
20 FOR N=1 TO 3
30 FOR J=1 TO 2
40 READ N$(N,J)
50 NEXT J:NEXT N
60 FOR N=1 TO 3
70 AT 5,N+3:PRINT N$(N,1);
80 AT 25,N+3:PRINT N$(N,2)
90 NEXT N
100 DATA "Caballo","Potro","Vaca","Ternero","Per
ro","Cachorro"
```

Fig. 5.17 Matriz de filas y columnas. Nótese la forma de dimensionar.

FOR ... NEXT. La lectura se realiza en las líneas 20 a 50. El bucle de las líneas 60 a 90 escribe los elementos de la matriz en columnas. En este bucle N es el número de fila; los números de columna son 1 y 2. Las filas contienen nombres de animales distintos. Por columnas se separan los nombres que se dan a los animales adultos y a sus crías, respectivamente. En este ejemplo hemos utilizado una *matriz literal*; pero también se pueden construir matrices numéricas, para las cuales, al dimensionarlas, no se especifica la longitud de los elementos.

El programa de la figura 5.18 es un poco más ambicioso. El objetivo es almacenar un conjunto de nombres y números de teléfono, que el programa capta por medio de las líneas 20 a 50. Una vez completa la matriz, se puede

```
10 CLS:DIM A$(50,2,20)
20 FOR N=1 TO 50
30 AT 2,8:INPUT"Nombre ";A$(N,1)
40 AT 2,10:INPUT"Tel. ";A$(N,2)
50 CLS:NEXT N
60 CLS:AT 12,8:PRINT"LISTA COMPLETA"
70 AT 2,10:INPUT"Elija una inicial ";J$
80 FOR N=1 TO 50
90 IF J$=A$(N,1)(1) THEN
100 PRINT"Nombre: ";A$(N,1)
110 PRINT"Tel.: ":A$(N,2)
120 END IF
130 NEXT N
```

Fig. 5.18 Utilización de una matriz literal en una sencilla aplicación de listín telefónico.

pedir el teléfono de una persona sin más que teclear su inicial. Para no hacer el programa demasiado largo, no hemos incluido trampas para errores, siempre necesarias en estos casos. Por ejemplo, como mínimo habría que prever lo siguiente:

```
PRINT"Lo siento; no encuentro ningún nombre que empiece por ";J$
```

Lo que hay de nuevo para nosotros en este programa está en la línea 90. Estamos tratando de identificar la primera letra de un nombre, y el nombre está en forma de lista. J$ es la inicial tecleada, y queremos averiguar si alguno de los nombres A$(1,1), A$(2,1), A$(3,1), ... empieza con esa letra. Para extraer la primera letra de un elemento de una lista hace falta una extraña expresión: A$(N,1)(1). Si nos paramos a pensarlo, esto es bastante lógico. Si A$(1) extrae la primera letra de A$, para hacer lo mismo con A$(N,1) tendremos que escribir A$(N,1)(1). Con la misma facilidad podríamos extraer dos letras: A$(N,1)(1 TO 2).

MISCELÁNEA

Vamos a examinar una función literal que aún no conocíamos: DATE$. Si tecleamos PRINT DATE$, veremos en la pantalla la fecha y hora siguientes:

1963 Apr 08 06:31:55

Estos datos se pueden modificar con la orden SDATE, seguida de varios números, separados por comas, que representan año, mes, día, hora, minutos y segundos. Por ejemplo, si se teclea

SDATE 1985,6,14,21,12,20

los datos asignados a DATE$ serán

1985 Jun 14 21:12:20

si bien, para cuando hayamos escrito PRINT DATE$, la hora habrá aumentado en unos cuantos segundos.

Hay una variable numérica, DATE, que controla la forma de escritura de DATE$. Sin embargo, DAY$ es más útil para la mayor parte de los usuarios. DAY$ da un día de la semana abreviado a tres letras; por ejemplo, Mon (lunes) o Sat (Sábado). Se puede modificar la hora, sin afectar a la fecha, con la instrucción ADATE seguida de un número. Por ejemplo, ADATE 36 hace avanzar el cronómetro 36 segundos, mientras que ADATE −45 lo retrasa 45 segundos. Según el manual provisional, el cronómetro dispondrá de una fuente de alimentación con batería que lo mantendrá en funcionamiento aun cuando se desconecte el ordenador. En mi QL no existe tal batería; si

la incorporaran en versiones posteriores, ADATE serviría para adelantar o atrasar una hora el reloj en verano (con ADATE 3600 o ADATE −3600).

Otra instrucción útil es INSTR. Sirve para determinar si una cadena literal está contenida en otra. La forma de utilizarla es

X = A$ INSTR B$

para averiguar si A$ está contenida en B$. Si lo está, X es la posición en B$ en la que se encuentra la primera letra de A$. Si no lo está, o si A$ es más larga que B$, X será cero. Por supuesto, si no se necesita más que ver el número, basta con teclear

PRINT A$ INSTR B$ (ENTER)

La figura 5.19 ilustra el funcionamiento de esta instrucción. Las líneas 20 a 40 asignan nombres a las cadenas literales. Las líneas 50 a 70 realizan las comprobaciones y escriben los resultados. Cuando los nombres están entre

```
10 CLS
20 a$="Albert Hall"
30 b$="Estatua de la Libertad"
40 c$="Torre de Pisa"
50 PRINT"En a$, ert esta en la posicion ";"ert"
INSTR a$
60 PRINT"En b$, ert esta en la posicion ";"ert"
INSTR b$
70 PRINT"En c$, ert esta en la posicion ";"ert"
INSTR c$
```

Fig. 5.19 Ejemplo de utilización de INSTR.

comillas, no es necesario dejar espacio entre las comillas y la palabra INSTR. En cambio, ese espacio es imprescindible cuando la cadena es una variable literal. Esta instrucción permite, por ejemplo, almacenar una lista de nombres en una cadena larga. Mediante INSTR se puede determinar si un nombre determinado está contenido en la cadena para, si no lo está, ponerlo al final de ella. Así se evitan todos los problemas de dimensionado de matrices y los que plantea la necesidad de prefijar la longitud de los elementos de la matriz. Esta intrucción debe recordarla el lector para utilizarla cuando esté en condiciones de escribir sus propios programas.

6

Menúes, subrutinas y programas

En la figura 4.14 introdujimos la idea de hacer una selección basada en la tecla que pulse el usuario. En aquel ejemplo la posibilidad de elección era limitada: S o N. Ahora podemos generalizar esa idea a un tipo de rutina que denominamos *menú*. Un menú es una lista de posibilidades de elección, generalmente funciones de un programa. Eligiendo una de esas posibilidades el usuario decide qué parte del programa se debe ejecutar. Una forma de hacer la oferta es numerar las posibilidades y pedir al usuario que teclee el número correspondiente a la opción deseada. En la figura 6.1 se muestra cómo

MENU

1. Hacer una lista de nombres nuevos.
2. Añadir nombres a la lista.
3. Leer nombres de la lista.
4. Borrar nombres de la lista.
5. Elegir un nombre.
6. Fin del programa.

ELIJA UN NÚMERO

Fig. 6.1 Un menú típico.

quedaría un menú típico en la pantalla. En un ordenador provisto de un BASIC de la época de los visigodos tendríamos que utilizar una sucesión de instrucciones del tipo

```
IF K = 1 THEN 1000
IF K = 2 THEN 2000
etc.
```

Hay otro método más sencillo, basado en una instrucción particular del QL: SELECT ON. Hay dos formas de utilizar esta instrucción; las estudiaremos detalladamente, pues SELECT no está incluida en los demás dialectos de BASIC.

Para empezar, supongamos que queremos elegir entre cuatro posibilidades tecleando un número del 1 al 4. Lo primero que tenemos que conseguir es que no se acepten números extraños a este margen, tales como 0, 5, −10; es decir, tenemos que programar una trampa para errores. En segundo lugar, la captación del número mediante INPUT es algo incómoda, pues obliga a pulsar ENTER después de teclear el número. Así pues, utilizaremos INKEY$ para captar la respuesta. A continuación comprobaremos que el

```
10 CLS:AT 17,2:PRINT"MENU"
20 AT 2,4:PRINT"Elija un numero del 1 al 4"
30 REPeat selecc
40 REPeat tecla
50 K$=INKEY$
60 IF K$<>"" THEN
70 EXIT tecla
80 END IF
90 END REPeat tecla
100 k=K$
110 SELect k=1 TO 4:EXIT selecc
120 PRINT"Respuesta incorrecta"
130 PRINT"Teclee un numero del 1 al 4"
140 END REPeat selecc
150 SELect ON k
160 ON k=1
170 PRINT"ESTA ES LA PRIMERA ALTERNATIVA"
180 REMark
190 ON k=2
200 PRINT"ESTA ES LA SEGUNDA ALTERNATIVA"
210 REMark
220 ON k=3
230 PRINT"ESTA ES LA TERCERA ALTERNATIVA"
240 REMark
250 ON k=4
260 PRINT"ESTA ES LA CUARTA ALTERNATIVA"
270 REMark
280 END SELect
290 PRINT"FIN DEL PROGRAMA"
```

Fig. 6.2 Selección de las alternativas de un menú con SELECT.

número elegido está en el margen correcto, para lo cual sirve la instrucción SELECT, según se muestra en la figura 6.2. En este programa hay dos rutinas, una que capta la respuesta y otra que comprueba la validez. Dado que este programa es posiblemente más largo que los que el lector está acostumbrado a ver, lo analizaremos por secciones. El defecto del SuperBASIC del QL es que requiere más trabajo de teclado para obtener los mismos resultados.

Vamos a centrarnos en las líneas 10 a 140. Las líneas 10 y 20 escriben las instrucciones para el usuario. En las líneas 30 y 40 empiezan sendas rutinas. Cuando en un programa se encuentren, como aquí, dos bucles anidados, la mejor forma de analizarlos es empezar por el interno; en este ejemplo el bucle interno está programado en las líneas 40 a 90. Esta rutina no es más que un bucle INKEY$; en BASIC ordinario se escribe en una sola línea. En la

```
30 REPeat selecc
50 K$=INKEY$
60 IF K$="" THEN GO TO 50
100 k=K$
110 SELect k=1 TO 4:EXIT selecc
120 PRINT"Respuesta incorrecta"
130 PRINT""Teclee un número del 1 al 4"
140 END REPeat selecc
```

Fig. 6.3 Una forma más sencilla de utilizar INKEY$.

figura 6.3 damos esta misma sección del programa en BASIC más «antiguo». Teniendo en cuenta que funciona igual de bien y es mucho más fácil de entender, nos quedaremos con esta versión. Pero volvamos a la figura 6.2. Si el usuario pulsa una tecla, las líneas 60 y 70 hacen que el programa salte a la línea 100. El efecto de este bucle es, pues, esperar por el usuario.

En la línea 100 la instrucción k = K$ convierte un número en forma de cadena literal a un número en forma de variable numérica, k. En otras versiones de BASIC la conversión es k = VAL(K$), que tiene la desventaja de que se produce error cuando K$ es una letra en lugar de un dígito. La línea 110 es la que realiza la selección. La instrucción SELECT k = 1 TO 4 hace que el margen de valores de k sea de 1 a 4. Si k está en ese margen, se ejecuta la segunda parte de la línea, EXIT selecc. Es decir, si la tecla pulsada es 1, 2, 3 o 4, se ejecuta la línea 150. En cambio, si se pulsa una tecla literal (de letra), el programa se detiene con un mensaje de error en la instrucción k = K$ de la línea 100. Si la tecla pulsada es numérica, pero fuera del margen correcto, la instrucción SELECT de 110 permite que se ejecute 120 y el

programa informa al usuario de su error. Así se llega a 140, donde END REPEAT reconduce el programa a la línea 30. El principal defecto de este programa, por lo demás aceptable, está en el mensaje de error que se produce cuando se pulsa una tecla literal. Esto no se puede obviar con una instrucción tal como SELECT K$ = "1" TO "4", ya que la SELECT requiere que la variable sea numérica. Cuando se utiliza INKEY$ para detectar la pulsación de una tecla numérica, es preferible realizar la comprobación con CODE (IF CODE(K$) > 52 THEN...).

Supongamos que ya tenemos un valor de k correcto. Entonces podemos realizar la selección (líneas 150 a 290). La instrucción SELECT ON k bifurca el programa hacia la sección en que se mencione el valor de k. Las diversas secciones están identificadas porque empiezan con ON k = ; cada sección puede contener todas las líneas que sean necesarias. En este ejemplo, como queríamos dejarlo lo más corto posible, nos hemos limitado a una instrucción PRINT por sección. Si el usuario teclea 1, la línea 150 envía el programa a 160, y se ejecutan todas las líneas contenidas entre ésta y la instrucción ON k = 2. Análogamente ocurre para las otras posibles respuestas del usuario. Cuando concluye la ejecución de una sección, el programa salta a la línea que sigue a END SELECT; en este caso, a la línea 290, donde hemos puesto el fin de programa. Aunque parezca innecesario, es buena idea hacer que se escriba en la pantalla FIN DEL PROGRAMA cuando éste termina. Hay muchos programas que no lo hacen; cuando el programa termina, el usuario se queda mirando a la pantalla, preguntándose qué tiene que hacer a continuación.

PROGRAMACIÓN MODULAR

Muchos programas empiezan con un título, unas instrucciones para el usuario y un menú. Dependiendo de la elección que se haga en el menú, la acción se reconduce a alguna parte del programa y más tarde éste termina o vuelve al menú. El método de selección basado en SELECT hace relativamente fácil escribir programas de este tipo.

Es importante que los programas estén formados por secciones independientes. Todas las versiones de BASIC permiten la programación de *subrutinas*, secciones que se pueden invocar (o llamar) desde cualquier lugar del programa. El QL y el BBC Micro son los dos únicos ordenadores que ofrecen un recurso de programación mucho más útil: el *procedimiento*. Dedicaremos el resto del capítulo a estudiar los procedimientos del QL.

Para empezar, ¿qué es un procedimiento? Alguien dijo una vez que le era imposible describir un elefante, pero que lo reconocería fácilmente si se encontrara con él en la calle. Un procedimiento es más difícil de describir que

de ilustrar con un ejemplo, de forma que vamos a mezclar la descripción con la ilustración para que el concepto quede claro. Un procedimiento es una orden que pide al ordenador que realice alguna acción y luego retorne para ejecutar la instrucción siguiente. En este sentido no es distinto de las demás órdenes. Cuando tecleamos PRINT A, el ordenador realiza la acción siguiente: prepara una línea nueva en la pantalla, busca el valor actual de una variable cuyo nombre sea «A» y escribe ese valor en la pantalla (si no se ha definido la variable «A», en la pantalla escribe un asterisco). Una vez concluida la acción, «retorna» y queda a la espera de la siguiente orden.

Esta acción es demasiado compleja como para poder ser ordenada con una sola palabra. En este caso, no bastaba con escribir PRINT; había que añadir la «A». Pues bien, la «A» era el *parámetro* de la instrucción, es decir, lo que había que añadir a la instrucción para que quedara completa. PRINT sin parámetro escribe una línea en blanco; con un parámetro, escribe el valor del parámetro. Hay instrucciones que requieren varios parámetros. Por ejemplo, AT necesita dos: el número de columna y el de fila.

En el QL es posible definir instrucciones. Una instrucción así definida es lo que llamamos «procedimiento». En algún lugar del programa se deben incluir ciertas líneas que expliquen al ordenador qué significa la nueva *palabra de instrucción* (el nombre que se da al procedimiento). Cumplida esta condición, el programa ya puede utilizar el procedimiento. Si éste necesita parámetros, tienen que ser suministrados en el orden correcto, pues de lo contrario se provocará el inevitable mensaje de error. Veamos un ejemplo.

```
1 rem 6-4
10 DEFine PROCedure centrar (A$)
20 princ=(38-LEN(A$))/2
30 CLS
40 AT princ,1:PRINT A$
50 END DEFine
60 centrar "ESTO ES UN TITULO"
```

Fig. 6.4 Ejemplo de definición y utilización de un procedimiento.

La figura 6.4 muestra un procedimiento muy sencillo. Las líneas 10 a 50 describen la acción que se le asigna. Al introducir la línea 10 hemos tecleado DEF PROC (si bien en el listado el ordenador escribe DEFine PROCedure) para indicar que estamos iniciando la definición de un procedimiento. A continuación escribimos el nombre del procedimiento, que es la palabra de instrucción que en adelante utilizaremos para invocarlo. Le hemos dado el nombre «centrar» porque la acción va a consistir en centrar una frase en la primera línea de la pantalla. Lo mismo que la instrucción PRINT, este pro-

cedimiento tendrá un parámetro, que será la cadena que queramos escribir centrada. En la definición los parámetros tienen que estar entre paréntesis. En cambio, cuando invoquemos el procedimiento los paréntesis no serán necesarios.

Concluida la línea de definición, tenemos que detallar en qué consiste la acción que se encomienda al procedimiento. La línea 20 define la variable «princ», que es el número de columna para la instrucción AT. La fórmula que hemos utilizado es la misma que en el capítulo 2 (adecuada para el televisor, pero no para el monitor). La línea 30 borra la pantalla; la línea 40 contiene las instrucciones AT y PRINT A$. Otra vez nos encontramos con el concepto de *variable local*; A$ es local al procedimiento. Esto quiere decir que A$ tiene significado en las líneas 10 a 50, pero no en otros lugares del programa. Este nombre de variable se puede utilizar en otros lugares del programa sin entrar en conflicto con la definición del procedimiento.

La línea 50 señala el final de la definición. En la línea 60 se prueba el funcionamiento del procedimiento recién definido escribiendo su nombre y un parámetro. El ordenador examina las líneas 10 a 50, asigna el parámetro a la variable A$ igual que si hubiéramos escrito

A$ = "ESTO ES UN TITULO"

y opera con esta variable según se especifica en la definición.

Hagamos una observación importante. En este ejemplo hemos definido el procedimiento al principio del programa; pero esto no tiene que ser así necesariamente. Las líneas de definición pueden estar en cualquier lugar del programa. Para comprobarlo, modifique los números de línea en el programa de la figura 6.4, de forma que DEF PROC esté en la línea 100 y END DEF en la 500. Entonces la línea 60 será la primera del programa. En ella se invoca un procedimiento que aparentemente aún no ha sido definido. Para el ordenador esto no es problema, porque siempre que encuentra una palabra de instrucción desconocida examina el programa entero para buscar su definición. Si no la encuentra, emite un mensaje de error.

PÓNGASE EN MARCHA

Para justificar la inversión en su nuevo QL, al lector quizá le baste con utilizar los programas que puede adquirir en el comercio. Más instructivo e interesante es que introduzca y pruebe los programas que se publican en las revistas especializadas, posiblemente modificándolos para adaptarlos a sus preferencias o a sus necesidades. Pero la máxima satisfacción la obtendrá, sin duda alguna, del diseño de sus propios programas. Éstos no tienen por qué ser obras maestras. Concebir un programa, escribirlo y comprobar que funciona ya es placer suficiente, sobre todo por el hecho de que es su propia

obra. Después de todo, comprar un ordenador y no programarlo es como comprar un Ferrari para que lo conduzca un chófer.

Algunos lectores querrán escribir programas que pongan orden en su colección de sellos o de discos, o para archivar recetas de cocina o datos técnicos de locomotoras de vapor antiguas. Los programas de este tipo se llaman *bases de datos*, pues su misión principal es recopilar, ordenar y archivar datos. Otros lectores estarán más interesados en juegos, manejo de colores, sonidos, dibujos u otros programas basados en los gráficos y en su movimiento. Estos programas utilizan las técnicas que describiremos en los capítulos 8 a 10. En esta sección vamos a describir las bases de datos, pues su diseño tiene utilidad en todos los programas. En cuanto el lector sea capaz de escribir programas de este tipo, estará preparado para progresar, utilizando los mismos métodos, hasta el diseño de programas gráficos y de sonido. Tenga en cuenta, no obstante, que el movimiento rápido de los gráficos no se puede programar en BASIC. La razón es que la lentitud de BASIC no permite dar realismo a los movimientos ni controlar numerosos objetos móviles. Los programas de juegos gráficos que puede adquirir en las tiendas están escritos en *código de máquina*, lo que significa que las instrucciones son un conjunto de números, que es lo que el microprocesador 68008 (corazón del ordenador) entiende directamente. La programación en código de máquina no utiliza para nada el BASIC, y es mucho más difícil. De hecho, muchos programas en código de máquina se elaboran con ayuda de otros programas en grandes ordenadores, y luego se los adapta para su utilización en ordenadores domésticos. No obstante, quien aprende a programar en BASIC llega a poder programar en código de máquina; sólo es cuestión de experiencia, muchísima experiencia.

Hagamos dos observaciones muy importantes. Primero, que la experiencia es decisiva en el diseño de programas. Si sus primeros esfuerzos los dirige a proyectos no ambiciosos, aprenderá mucho de ellos. En efecto, la probabilidad de tener éxito es tanto mayor cuanto más sencillo sea el programa. Se aprende más de un pequeño programa que funciona a la primera o tras unos pequeños retoques que de un gran programa que no se termina de depurar nunca. En segundo lugar, el diseño de un programa empieza por desconectar el ordenador, y preferiblemente a buena distancia de él. La razón es que el diseño es en principio un trabajo de planificación, y ésta es incompatible con la tentación que representa tener el teclado al lado.

LÁPIZ Y PAPEL

Empiece por preparar papel en abundancia. Yo lo utilizo de formato A4, y con agujeros que me permitan archivar las hojas. De esta forma las manten-

go ordenadas y puedo intercalar otras donde sea necesario. Observe que hablo de hojas, en plural. El diseño de un programa, por sencillo que parezca en principio, requerirá más de una. Si el programa es medianamente complejo, es fácil que antes de llegar al teclado haya llenado un par de docenas de hojas con notas de diseño y listas de instrucciones. Vamos a tomar un ejemplo de programa y a desarrollar su diseño. El programa es muy sencillo, pero ilustrativo.

Empezaremos por escribir qué queremos que el programa haga. Esto puede parecer superfluo, porque siempre creemos tener las ideas claras, pero no lo es. Se dice que los árboles impiden ver el bosque, y el dicho es muy cierto aplicado a la informática. Si se empieza a escribir directamente en BASIC, se llega a estar tan absorto con los detalles que al final se acaba por olvidar para qué sirve toda esa maraña de instrucciones. En cambio, si escribimos en un papel nuestro objetivo, tendremos una meta a la que llegar, y esto es tan importante en el diseño de programas como en la vida. Y no basta con escribir unas cuantas palabras; el problema tiene que ser desmenuzado en todo detalle, pues de no hacerlo así no podremos programarlo. Por otra parte, un análisis suficientemente detallado hace posible el diseño de un programa bien *estructurado*; esto significa que las diversas secciones del programa forman una sucesión lógica, fácilmente susceptible de ampliación y modificación. Si el lector aprende a programar con esta disciplina, sus programas serán fáciles de entender, tardarán menos en funcionar y podrán ser ampliados para que realicen más funciones que las previstas en las primeras etapas de su concepción. El QL facilita la escritura de programas estructurados.

Consideremos, como ejemplo, la figura 6.5. Esto es un esbozo de la programación de un juego educativo sencillo. El plan del programa indica qué se espera del juego. Tiene que escribir en la pantalla una palabra en caste-

---

Objetivos

1. Escribir una palabra en castellano en la pantalla.
2. Preguntar cómo se escribe esa palabra en inglés.
3. La respuesta tiene que ser correcta, incluso en ortografía.
4. El usuario no debe poder leer las respuestas en un listado del programa.
5. Dar un punto por cada respuesta acertada.
6. Permitir un segundo intento en cada pregunta.
7. Llevar la cuenta del número de intentos.
8. Presentar como resultado final el número de aciertos y el de intentos.
9. Elegir las palabras propuestas aleatoriamente.

Fig. 6.5 Esbozo del plan de un programa (¡de su primer programa!).

llano, tomada al azar de una lista, y preguntar cómo se escribe en inglés. Además, tiene que comprobar la ortografía de la respuesta. También hay que evitar que el usuario, posiblemente familiar del programador, averigüe las respuestas haciendo un listado del programa y buscando las líneas de DATA. Todo juego tiene que disponer de un sistema de tanteo; vamos a dar un punto por cada palabra acertada. Puesto que la ortografía ha de ser correcta, quizá haya que dar más de una oportunidad en cada pregunta. Finalmente, hay que llevar la cuenta no sólo de los aciertos, sino también de los intentos, para informar al final de cada juego. Y esto es suficiente, a no ser que queramos complicar más el juego. Para ser un primer programa, no está nada mal. ¿Qué tenemos que hacer ahora?

---

1. Escribir en la pantalla el título y las instrucciones del juego.
2. Preparar variables, leer datos, etc.
3. Repetir hasta el punto 8.
4. Elegir número aleatorio.
5. Hacer una jugada.
6. Actualizar y escribir tanteador.
7. Preguntar al usuario si quiere seguir jugando.
8. Si el usuario responde «S», ir a 3.
9. Indicar que el juego ha terminado.

Fig. 6.6 Siguiente etapa de desarrollo del plan.

La respuesta es la siguiente: tenemos que actuar de la misma manera que un pintor hace un cuadro o un arquitecto diseña una casa, es decir, ocupándonos primero de los esquemas y después de los detalles. Los esquemas de este programa son las etapas que forman el conjunto de acciones. Es evidente que tendremos que definir variables, dimensionar listas y hacer otros preparativos similares. Después tendremos que desarrollar el juego. Luego tendremos que ocuparnos de los contadores de aciertos e intentos. En la figura 6.6 se indica qué es lo que deberíamos tener escrito en un papel a estas alturas.

LOS CIMIENTOS

Ahora ya podemos escribir unas líneas de programa, si bien sólo las fundamentales. Lo que no podemos hacer de ningún modo en esta etapa es ponernos a llenar hojas y más hojas con líneas de BASIC. Como es bien sabido,

los cimientos de toda obra son decisivos, y tienen que estar bien echados antes de empezar a levantar las paredes.

En la figura 6.7 se muestra la primera versión del programa. Consta de solamente catorce líneas, todo lo que necesitamos por ahora. Recuerde que estos son los cimientos, no el edificio. Además, no debemos preocuparnos por los números de línea que utilicemos, ya que en cualquier momento podemos renumerar el programa. Lo que sí debemos hacer es dar a los procedimientos nombres que más tarde tengan algún significado para nosotros.

```
10 CLS:CLEAR:RESTORE
20 titulo
30 instrucciones
40 prep
50 REPeat partida
55 selecc
60 juego
70 tanteo
80 otravez
90 IF resp$="N" OR resp$="n" THEN
100 EXIT partida
105 END IF
110 END REPeat partida
120 PRINT"FIN DEL JUEGO"
```

Fig. 6.7 Cimientos del programa ejemplo.

Pero volvamos al programa. Como se puede observar, consiste en un conjunto de procedimientos que aún no hemos desarrollado. El programa sigue el esquema de la figura 6.6. La única línea que no forma parte de un procedimiento es el IF de la línea 90. Esta línea comprueba la respuesta del usuario a la pregunta «¿Quiere seguir jugando?». Para captar la respuesta tendremos que escribir una subrutina con INKEY$.

Estudie detenidamente estas catorce líneas porque son muy importantes. Al utilizar procedimientos, podemos comprobar fácilmente esta versión del programa; después de todo, no hay mucho en lo que podamos habernos equivocado. Ahora debemos decidir en qué orden vamos a escribir los procedimientos. El peor suele ser el orden en que aparecen en el programa. Las instrucciones para el usuario y la preparación de la pantalla se deben dejar para el final, pues son la parte del programa que menos afecta al funcionamiento global. Si las escribimos al principio, siempre cabe la posibilidad de

que más tarde introduzcamos en el programa mejoras que nos obliguen a escribirlas otra vez. En este momento es buena idea escribir

11 GO TO 40

con lo que puenteamos todo lo referente a título e instrucciones. Esto sirve también para ahorrar tiempo cuando se está probando el programa. Otra advertencia sobre las peculiaridades del QL, espero que solamente de las primeras versiones: cuando me puse a escribir instrucciones en forma de una larga instrucción PRINT, con signos de barra inclinada hacia la izquierda para forzar el salto a la línea siguiente, el programa no quedaba correctamente grabado. Perdí varias copias del programa antes de descubrir la causa del problema. *Siempre* es conveniente grabar los programas por etapas, dando a cada nueva versión un nombre diferente. De esta forma, si el QL pierde el control de lo que está haciendo, y esto es algo que me ha ocurrido con demasiada frecuencia, la pérdida no es tan grande porque se puede volver a cargar una versión anterior del programa sin necesidad de tener que teclearlo todo de nuevo.

El siguiente paso es ponernos al teclado (¡por fin!) e introducir el programa tal como lo tenemos. No podemos probarlo mientras no hayamos desarrollado los procedimientos, pero sí debemos grabar este núcleo del programa para luego ir añadiéndole y comprobando procedimientos. En cada etapa, una vez comprobado el funcionamiento de lo que se haya añadido, lo volveremos a grabar (con nombre diferente), y así dispondremos siempre de la versión más moderna. Estas comprobaciones parciales son muy importantes. Si las aplazáramos para cuando el programa fuera demasiado largo, el trabajo de depuración llegaría a ser tedioso y quizá imposible. En cambio, si vamos probando cada procedimiento después de escribirlo, siempre podremos «fiarnos» del trabajo que ya llevamos hecho y concentrarnos en los errores del último añadido. Recuerde que para ejecutar un procedimiento basta con teclear su nombre y pulsar ENTER.

LA ETAPA DE LOS PROCEDIMIENTOS

Lo que nos queda por hacer, y no es poco, es escribir los procedimientos. Para algunos no tenemos que pensarlo mucho. Veamos, por ejemplo, el procedimiento que hemos denominado «otravez». Se trata de la conocida rutina INKEY$, junto con algún PRINT. En la figura 6.8 se da una posible versión. Este procedimiento es muy sencillo, y por eso hemos empezado por él. Introdúzcalo y compruebe el funcionamiento del programa resultante.

Vamos a abordar lo que el lector seguramente considerará más difícil: los procedimientos incluidos en el bucle «partida». Pues bien, para escribirlos

```
1000 DEFine PROCedure otravez
1005 PRINT:PRINT"Quiere seguir jugando (S/N)?"
1010 REPeat pregunta
1020 K$=INKEY$
1030 IF K$<>"" THEN
1040 resp$=K$
1050 EXIT pregunta
1060 END IF
1070 END REPeat pregunta
1080 END DEFine
```

Fig. 6.8 Procedimiento «otravez».

no tiene que aprender nada nuevo. Estos procedimientos se escriben por el mismo método que el núcleo del programa; es decir, escribiendo detalladamente las funciones que queremos que realice y ordenando las etapas en que se desarrollará la acción. Si encontramos algo para lo que no tengamos solución inmediata, podemos relegarlo a un nuevo procedimiento.

---

1. Escribir en la pantalla una palabra en castellano; preguntar cómo se escribe esa palabra en inglés.
2. Almacenar las palabras propuestas en una lista literal.
3. Almacenar las respuestas en forma de códigos ASCII en líneas de DATA (una respuesta en cada línea).
4. Utilizar las siguientes variables: «intento» para el contador de preguntas; «puntos» para el tanteador; «oport» para el número de respuestas dadas en cada pregunta.
5. Utilizar P$ como nombre de la lista de las palabras propuestas.

Fig. 6.9 Planificación del procedimiento «juego».

Consideremos la figura 6.9, en la que hemos hecho un plan para el procedimiento «juego» y anotado información que necesitaremos en los procedimientos «prep» y «selecc». Antes de que comience el procedimiento «juego» tenemos que poner a cero la variable «oport». Esta variable nos servirá para decidir si hay que dar otra oportunidad al jugador. También necesitamos generar un número aleatorio. Estas dos acciones las realiza el procedimiento «selecc», el cual tiene que ser independiente de «juego» porque, si el jugador da una respuesta errónea en el primer intento, hay que darle otra oportunidad, no pasar a la siguiente pregunta.

La primera acción de «juego» es escribir una palabra en castellano y preguntar cómo se escribe en inglés. Esa palabra tendrá que haber sido elegida mediante un número aleatorio entregado por el procedimiento «selecc». Las palabras en castellano las guardamos en una lista. Esto tiene varias ventajas. La primera es la facilidad con que podemos seleccionarlas aleatoriamente. Otra es que también facilita emparejar las preguntas con las respuestas. Si las preguntas son elementos de una lista, con índices de 1 al 10, podemos poner la respuestas en líneas DATA (un conjunto de códigos en cada una) y leerlas sirviéndonos de la instrucción RESTORE.

Recordará el lector que no queríamos que el usuario pudiera ver las respuestas en la línea DATA si se le ocurría listar el programa. La solución más sencilla es ocultarlas en forma de códigos ASCII; el jugador que quiera hacer trampa podrá descifrarlas, pero no sin trabajo. Hemos decidido poner las respuestas en líneas DATA haciendo que el primer dato sea el número de caracteres de la respuesta en cuestión. De esta forma, podemos hacer que RESTORE «apunte» a una línea determinada, leer el número de letras y luego formar un bucle FOR ... NEXT que lea los códigos de las letras. Cada código se convierte en una letra mediante CHR$, y las letras sucesivas se van añadiendo a la izquierda de la respuesta, A$.

También tenemos que decidir qué nombres vamos a dar a las variables. Es conveniente asignar nombres que estén relacionados con las magnitudes que representan. En este caso, «puntos» para el tanteo e «intento» para el número de respuestas son nombres adecuados. Hemos elegido «puntos» porque «tanteo» es el nombre de un procedimiento, y no se puede utilizar el nombre de un procedimiento como nombre de variable. La otra variable, «oport», indica si estamos en la primera o en la segunda oportunidad, de las dos que se dan al jugador con cada pregunta. Finalmente, para la lista de palabras propuestas elegimos P$.

EMPIEZA EL JUEGO

En la figura 6.10 se muestra el procedimiento «selecc», y en la 6.11, el «juego» que ha resultado del plan de la figura 6.9. La acción consiste en generar un número aleatorio, utilizarlo para elegir y escribir una palabra y cap-

```
1090 DEFine PROCedure selecc
1095 oport=0:v=RND(1,10)
1097 END DEFine
```

Fig. 6.10 Procedimiento «selecc».

```
1100 DEFine PROCedure juego
1120 CLS:AT 2,2:PRINT"La palabra en castellano e
s: ";P$(v)
1130 AT 2,4:PRINT"En inglés se dice: ";
1140 INPUT X$:intento=intento+1
1150 A$="":RESTORE 6000+v
1160 READ longitud:FOR J=1 TO longitud
1170 READ num:A$=A$ & CHR$(num)
1175 NEXT J
1180 END DEFine
```

Fig. 6.11 Definición del procedimiento «juego».

tar la respuesta. Y eso es todo, porque la comprobación de la respuesta y el mantenimiento del contador se confían a otro procedimiento. Siempre es conveniente desmenuzar las tareas al máximo, para no tener que escribir grandes trozos de programa de una vez.

El procedimiento empieza en la línea 1100. En la línea 1120 escribimos la palabra correspondiente al número aleatorio generado y a continuación pedimos al jugador su respuesta. La segunda instrucción de 1140 cuenta el número de intentos. Aquí es precisamente donde hay que poner ese contador, pues debemos incrementar el número de intentos cada vez que se produzca una respuesta. Ahora tenemos que asignar a A$ la cadena vacía; si no lo hiciéramos, las sucesivas respuestas se irían acumulando a la derecha de A$. La instrucción RESTORE 6000 + v selecciona la línea de DATA de la que vamos a leer datos. Por ejemplo, si v es 3, la palabra propuesta está en la cadena P$(3), y la respuesta en la línea 6003. Leemos el primer dato de la lista, que es la «longitud» de la respuesta. El bucle de las líneas 1160 a 1175 lee los restantes números, los convierte en caracteres y los añade a la derecha de A$ hasta formar la respuesta completa. Todo lo que tenemos que hacer ahora es comparar esta respuesta con la captada por INPUT en la línea 1140. Pero esta acción se la encomendamos al procedimiento «tanteo».

MANTENIMIENTO DEL TANTEADOR

Con el procedimiento «juego» grabado y a salvo en el microdrive, podemos empezar a pensar en otro procedimiento: «tanteo». Como ya sabemos, hay que empezar por planificarlo (Fig. 6.12). Cada vez que el usuario dé una respuesta correcta, hay que incrementar la variable «puntos» y volver al programa principal. En cambio, si la respuesta es incorrecta, hay que escribir un mensaje y dar otra oportunidad. En este caso tenemos que decrementar

1. Si la respuesta es correcta, incrementar «puntos».
2. Si la respuesta es incorrecta y oport = 0, dar otra oportunidad y hacer oport = 1.
3. Si la respuesta es incorrecta y oport = 1, hace oport = 0 y pasar a la siguiente pregunta.
4. Emitir sonidos diferentes en cada caso (véase el capítulo 10).

Fig. 6.12 Planificación del procedimiento «tanteo».

«intento» para no contar la pregunta dos veces. Si la segunda respuesta tampoco es correcta, ya no hay más oportunidades. En ese punto debemos incluir algún sonido. Podríamos programar pitidos diferentes para la respuesta correcta y para el primer error y el segundo. Póngalo todo en el plan.

```
1200 DEFine PROCedure tanteo
1210 REPeat pruebas
1220 IF X$=A$ THEN
1230 puntos=puntos+1
1240 AT 2,6:PRINT"Correcto. Su puntuacion es aho
ra ";puntos
1245 PRINT" en ";intento;" intentos."
1250 BEEP 20000,11
1260 EXIT pruebas
1270 ELSE IF oport=1 THEN
1280 oport=0
1290 AT 2,6:PRINT"No ha habido suerte. Otra vez
sera."
1292 BEEP 15000,100
1300 PAUSE 100
1310 EXIT pruebas
1320 ELSE IF oport=0 THEN
1330 AT 2,6:PRINT"No es correcto, pero le doy"\"
otra oportunidad."
1335 BEEP 10000,80
1340 oport=1:intento=intento-1
1350 PAUSE 100
1360 juego
1370 END IF
1380 END REPeat pruebas
1390 END DEFine
```

Fig. 6.13 Definición del procedimiento «tanteo».

En la figura 6.13 se muestra el procedimiento «tanteo» que ha resultado de este plan. Está incluido en un bucle REPEAT para permitir la segunda oportunidad. Las líneas 1220 a 1260 actúan cuando la respuesta es correcta. Si no lo es, la línea 1270 comprueba si estamos en la segunda oportunidad; en caso afirmativo se ejecutan las líneas 1280 a 1310. Si la respuesta es incorrecta pero la oportunidad es la primera, el procedimiento pasa a la línea 1320. Ésta escribe el mensaje; a continuación se anota que ha terminado la primer oportunidad y se ofrece al jugador la segunda. Esto se consigue repitiendo la llamada a «juego» (línea 1360). Cuando el jugador responde, el procedimiento se repite (gracias al END REPEAT pruebas) para comprobar la respuesta.

Ahora que estamos en el buen camino, podemos empezar a pulir el resto

```
1500 DEFine PROCedure prep
1510 intento=0:puntos=0:oport=0
1520 DIM P$(10,20)
1530 RESTORE:FOR J=1 TO 10
1540 READ P$(J):NEXT J
1550 END DEFine
```

Fig. 6.14 Procedimiento «prep» que inicializa las variables y dimensiona la lista.

```
2000 DEFine PROCedure instrucciones
2010 CLS:AT 13,1:PRINT"INSTRUCCIONES"
2020 AT 1,3:PRINT"El ordenador le propondra una
palabra"
2025 PRINT"en castellano. Usted debe escribirla"
2030 PRINT"en ingles, cuidando la ortografia y"
2035 PRINT"poniendo la primera letra en"
2040 PRINT"mayuscula. El ordenador llevara la"
2045 PRINT"cuenta de los aciertos. Tendra dos"
2050 PRINT"oportunidades para cada pregunta."
2055 PRINT"PULSE LA BARRA ESPACIADORA"
2060 PRINT"PARA EMPEZAR."
2065 REPeat bucle
2070 IF KEYROW(1)=64 THEN
2075 EXIT bucle
2076 END IF
2080 END REPeat bucle
2090 END DEFine
```

Fig. 6.15 Procedimiento «instrucciones». Deje siempre este procedimiento para el final.

de los procedimientos. En la figura 6.14 se muestra el procedimiento «prep», que se encarga de dimensionar las listas e inicializar las variables. La línea 1510 prepara las variables del tanteador. La línea 1520 dimensiona la lista P$; las líneas 1530 y 1540 leen las palabras de una lista DATA y las asignan a P$. Como de costumbre, dejaremos las listas de datos para el final.

La figura 6.15 es el procedimiento que escribe las instrucciones para el usuario. La figura 6.16 da el listado del procedimiento «título». En la línea del título hemos incluido una pausa; las letras están escritas con caracteres de tamaño CSIZE 3, 1. Esto es muy llamativo y recomendable, siempre que

```
3000 DEFine PROCedure titulo
3010 CLS:CSIZE 3,1
3020 AT 7,1:PRINT"VOCABULARIO INGLES"
3030 PAUSE 300
3040 CSIZE 2,0
3050 END DEFINE
```

Fig. 6.16 Procedimiento para escribir el título en caracteres grandes.

el texto quepa en la pantalla. Las instrucciones están en líneas separadas porque, como dije antes, una instrucción larga trastornaba de tal forma la memoria de mi QL que el programa no se podía grabar ni reproducir correctamente. Finalmente, en la figura 6.17 se pueden ver las líneas de datos.

```
6000 DATA "Perro","Gato","Vaca","Caballo","Galli
na","Zorro","Canguro","Ganso","Leon","Cerdo"
6001 DATA 3,68,111,103
6002 DATA 3,67,97,116
6003 DATA 3,67,111,119
6004 DATA 5,72,111,114,115,101
6005 DATA 3,72,101,110
6006 DATA 3,70,111,120
6007 DATA 8,75,97,110,103,97,114,111,111
6008 DATA 5,71,111,111,115,101
6009 DATA 4,76,105,111,110
6010 DATA 3,90,105,103
```

Fig. 6.17 Líneas de datos para preguntas y respuestas.

Ahora podemos reunir todos los elementos y probar el programa. Puesto que el programa ha sido diseñado por módulos, será fácil modificarlo. Por ejemplo, se pueden modificar las líneas DATA, o poner más palabras (no olvide entonces modificar la instrucción DIM de la línea 1520). Puede convertir este juego en cualquier otro que funcione a base de preguntas y respuestas sin más que cambiar los datos y las instrucciones. Puede incluir efectos sonoros espectaculares y gráficos que hagan el programa más atractivo. Uno de los defectos de este programa es que, tal como está, puede proponer la misma pregunta dos veces. Para subsanarlo haga lo siguiente: intercambie la palabra elegida con la última y luego restrinja el margen de los números aleatorios. Por ejemplo, si el primer número generado es el 5, intercambiamos la palabra número 5 con la número 10, y en la siguiente vuelta generamos un número aleatorio comprendido entre 1 y 9. Para ello, en lugar de la instrucción RND (1, 10) tendremos que poner RND(1,D), donde D parte de un valor inicial igual a 10 y es decrementada cada vez que el jugador da una respuesta acertada.

Desde luego, es mucho lo que se puede hacer para convertir este programa en un juego verdaderamente interesante. Lo hemos utilizado como ejemplo para demostrar que el lector ya está en condiciones de empezar a programar. Considérelo como un «juego de bloques» que puede reconstruir en la forma que más le guste. Como mínimo, le dará una idea de lo que podrá conseguir cuando conozca a fondo su QL. A medida que vaya creciendo su experiencia, podrá abordar proyectos más largos y complejos. Para entonces sabrá más acerca del QL de lo que se puede escribir en un libro.

7

Canales, microdrives y archivo de datos

El QL, a diferencia de la mayor parte de los ordenadores, tiene incorporado un sistema de almacenamiento de programas y datos. Mientras los demás ordenadores utilizan magnetófonos de cassette corrientes o discos flexibles, el QL dispone de dos «microdrives», que son una especie de cinta sin fin. Es de suponer que el lector, cualquiera que sea su experiencia anterior con ordenadores, no conozca este sistema de archivo de datos; por ello vamos a dedicar este capítulo al estudio de la grabación y lectura de datos en el QL. En el capítulo 1 ya hemos descrito el manejo del microdrive en lo referente a la grabación y carga de programas.

La terminología extraña, con palabras tales como *canales* y *dispositivos* es la única dificultad que se puede atravesar en nuestro camino. En cuanto el lector asimile el significado de estos términos, la utilización del microdrive empezará a parecerle sencilla e interesante, y podrá sacar mayor provecho de su ordenador. Expliquemos, pues, el significado de esas palabras.

Un *dispositivo* es, en este contexto, un aparato que recibe o entrega datos. El teclado, por ejemplo, es un dispositivo, pues envía señales eléctricas al ordenador cada vez que se pulsa una tecla. La pantalla es otro dispositivo, porque recibe del ordenador señales eléctricas que convierte en imágenes. El teclado es un dispositivo emisor: envía señales. La pantalla, en cambio, es un dispositivo receptor: recibe señales. En realidad, la pantalla del QL se comporta como si fuera varios dispositivos integrados en uno solo, pues se la puede dividir en tres áreas a las que se accede independientemente. Más adelante, en el capítulo 8, veremos cómo hacerlo.

Hay otros dispositivos que pueden operar como receptores o como emisores. Una *consola*, que es la combinación de teclado y pantalla, emite y recibe datos. También el microdrive es un dispositivo bidireccional. Se pueden interconectar varios ordenadores entre sí, de forma que en la red resultante cada QL puede ser emisor o receptor. En las instrucciones y órdenes que hacen

referencia a estos dispositivos empleamos abreviaturas tales como SCR para la pantalla y MDV para los microdrives.

Hemos dicho que las señales eléctricas pasan de un dispositivo a otro y, efectivamente, esa transferencia de señales es una de las misiones del ordenador. Pero resulta mucho más útil estudiar, no las señales en sí, sino lo que ellas representan. Cada conjunto de señales representa una unidad de datos denominada *byte*; los *datos* son la materia prima con la que operan los ordenadores. Los datos pueden ser números o letras: cualquier cosa que el ordenador sea capaz de manejar. Si usted ha escrito un programa que almacena y ordena los nombres y los cumpleaños de sus amigos por orden de fechas, para que el programa funcione habrá que suministrarle datos. En este ejemplo los datos serán los conjuntos de nombres y fechas de nacimiento. Si el programa está diseñado para archivar recetas de cocina, los datos serán las instrucciones de las recetas, los nombres de los platos y las cantidades de los ingredientes. Todo ordenador del que se pretenda que sirva para algo más que para juegos sencillos tiene que ser capaz de grabar y leer datos de este tipo, independientemente del programa que los genera o los utiliza.

El QL utiliza su memoria para diversos propósitos, muchos de los cuales escapan al control del usuario. Un programa muy largo que capte los datos y también los utilice puede no caber en la memoria. Mucho más práctico es disponer de un programa corto que reúna los datos con instrucciones INPUT y los grabe en microdrive. De esta forma los datos estarán a salvo de cualquier accidente que pueda corromper la memoria del ordenador. Después tendrá que haber otro programa que manipule esos mismos datos. La cantidad de información que puede residir en la memoria en un momento dado es mucho mayor si los dos programas son independientes que si están reunidos en uno solo.

Pero ¿qué tiene esto que ver con canales y dispositivos? Como ya sabemos, los dispositivos son elementos del sistema que reciben y emiten datos. Los *canales* son las vías de transmisión. Meditemos un poco sobre el funcionamiento del QL: cuando usted pulsa una tecla, algo aparece en la pantalla; el teclado es un dispositivo, la pantalla es otro y hay un canal que los une. Dicho de otro modo, hay un soporte físico por el que se transmiten señales eléctricas del teclado a la pantalla. Lo que importa al programador es que esos canales pueden ser controlados, en el sentido de que podemos abrir y cerrar canales a nuestro antojo. Por supuesto, no sería lógico cerrar todos los canales, salvo en aplicaciones muy especiales. Lo habitual es que se mantenga abierto el canal teclado-pantalla; sin embargo, si un programa requiere que el usuario teclee una palabra clave sin permitir que otra persona la vea en la pantalla, sería necesario cerrar ese canal.

Como el lector ya sospecha, hay canales que se abren en el momento de encender el ordenador. Esto es evidente en el caso del canal teclado-pantalla.

En realidad, este canal es múltiple. En efecto, cuando se pulsa una tecla la letra correspondiente aparece en la parte inferior de la pantalla, mientras que las instrucciones PRINT envían los textos a la parte superior. El QL maneja hasta 16 canales. Cuatro de ellos tienen misiones específicas que el programador normalmente no modifica. Los canales se identifican por el número del dispositivo, precedido del signo norteamericano de «número», #.

Como suele ocurrir en informática, los canales no están numerados del 1 al 16, sino del 0 al 15. Los canales #0 a #2 controlan las tres regiones en que se divide la pantala. Por ejemplo, la orden

PRINT #0, «prueba»

escribe la palabra «prueba» en la última línea de la pantalla. PRINT #1 escribe en la región en la que normalmente lo hacen las instrucciones PRINT. Finalmente, PRINT #2 escribe en la región habitualmente reservada para los listados; además, si se tiene conectado un televisor, en el canal #2 se escribe con letras rojas sobre fondo blanco.

Los canales #0 a #2 son controlados automáticamente por el QL cuando éste envía textos a la pantalla. Por ejemplo, una instrucción PRINT en la que no se especifique número de canal selecciona el canal #1. Si desea escribir en otra zona de la pantalla, ya sabe cómo hacerlo. Los canales #3 a #15 están normalmente libres. Aparentemente, el canal #3 es utilizado en ocasiones cuando están funcionando los microdrives, pero el QL lo vuelve a poner a disposición del usuario enseguida. Estos canales pueden ser utilizados por el usuario según convenga en cada programa. Para ello es necesario saber dos cosas: primero, cómo seleccionar un canal; segundo, cómo asignarlo a un dispositivo determinado.

Para seleccionar un canal se utiliza la instrucción OPEN. Esta palabra va seguida del signo # y del número del canal. Por ejemplo, OPEN #3,SER1 prepara el canal número 3 para enviar datos a la impresora a través de la puerta serie (véase el apéndice B). Ahora vamos a estudiar el manejo de los canales que se utilizan para grabar datos en los microdrives y para leerlos.

TÉCNICAS DE ARCHIVOS DE DATOS

¿Qué es un fichero?

El termino *fichero* aparece reiteradamente en este libro. Un fichero es un conjunto de caracteres lógicamente interrelacionados entre sí. Por ejemplo, los caracteres de que consta un programa en BASIC constituyen un fichero, ya que la relación que hay entre ellos es tal que el programa no funciona si falta alguno. Un conjunto de nombres y direcciones en código ASCII es un

	FICHERO DE AMIGOS
REGISTRO 1	
CAMPO 1	Nombre 1
CAMPO 2	Dirección 1
CAMPO 3	N.º de teléfono 1
CAMPO 4	Cumpleaños 1
REGISTRO 2	
CAMPO 1	Nombre 2
CAMPO 2	Dirección 2
CAMPO 3	N.º de teléfono 2
CAMPO 4	Cumpleaños 2
REGISTRO 3	
etc.	

Fig. 7.1 Significado de los término *registro* y *campo*.

fichero, porque forma un bloque de información relativa a un ente homogéneo; por ejemplo, al conjunto de nuestros amigos, clientes o acreedores. Otros ejemplos de ficheros son un conjunto de bytes en código de máquina y el conjunto de datos numéricos que utiliza un programa de análisis financiero. No obstante, en este libro consideraremos que un fichero es un conjunto de datos cualesquiera que podemos grabar en una cinta separadamente del programa que los va a utilizar. Así, un programa de contabilidad doméstica requerirá un fichero de conceptos y de cantidades de dinero. Este fichero será generado y actualizado por un programa, y habrá que archivarlo para poder ponerlo a disposición del programa en la próxima ejecución. Por poner otro ejemplo, supongamos que usted ha diseñado un programa para ordenar y controlar su colección de discos de 78 rpm. El programa requerirá, sobre todo al principio, la introducción de gran cantidad de datos acerca de las grabaciones. Esta información constituye un fichero y en algún punto del programa tendrá que grabarla. ¿Por qué? Pues por la sencilla razón de que, para que el programa sea verdaderamente útil, tendrá que ser capaz de manejar una cantidad de información mayor que la que cabe en la memoria del ordenador. De esto es de lo que trata este capítulo: de grabar la información con la que trabaja un programa; dicho de otra forma, de *archivar* la información.

No podemos hablar de métodos de archivo sin antes conocer la terminología necesaria. Los términos más importantes son los de *registro* y *campo* (Fig. 7.1). Un registro es un conjunto de datos referidos a un mismo elemento del fichero; es el concepto equivalente al de *ficha* en un fichero tradicio-

nal. Por ejemplo, en un fichero de locomotoras de vapor, a cada modelo se le dedicaría un registro. Dentro de este registro o ficha se reservaría espacio para cada una de las especificaciones de la máquina. Cada una de esas especificaciones ocupa un campo. Por ejemplo, un campo sería el año de fabricación, otro el nombre del fabricante, etc. El conjunto de todos estos campos constituye un registro, y el conjunto de todos los registros forma el fichero.

SISTEMA DE ARCHIVO EN MICRODRIVE

En este libro, dado que disponemos del sistema de microdrives del QL, ignoraremos los métodos de archivo que se basan en las líneas DATA de un programa de BASIC. Para algunos lectores el microdrive será el único sistema de grabación que conozcan; otros habrán utilizado magnetófonos de cas-

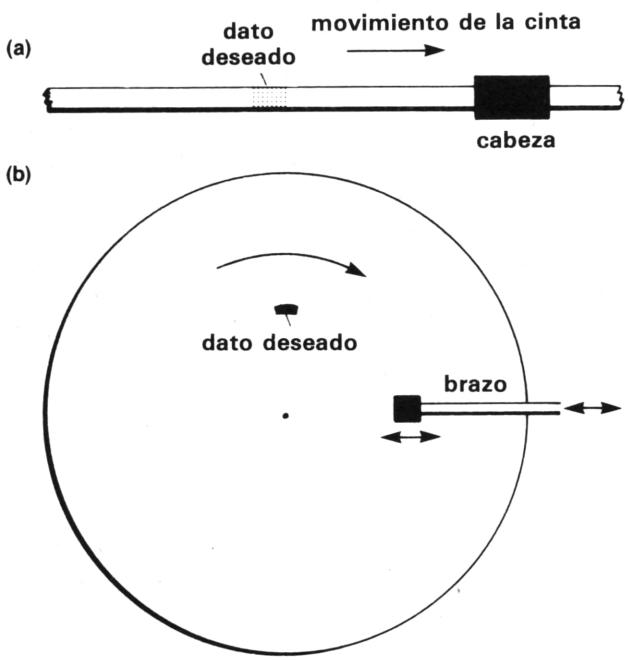

Fig. 7.2 Diferencia entre (*a*) ficheros secuenciales en cinta y (*b*) ficheros de acceso aleatorio en disco. Todos los puntos del disco son accesibles inmediatamente sin más que colocar la cabeza a la distancia del centro correcta. En cambio, para leer un dato de la cinta, hay que recorrer la cinta desde el principio del fichero hasta el lugar donde se encuentre el dato.

ssette. Si todo esto le resulta conocido, tenga paciencia; ya llegaremos a algo que usted no haya visto aún.

Hay dos tipos generales de ficheros y sólo uno de ellos es posible en la cinta de microdrive. Los dos tipos son los *ficheros secuenciales* y los *ficheros de acceso aleatorio*. La diferencia es sencilla, pero decisiva. En un fichero secuencial, la información se graba ordenadamente, un dato tras otro. No hay forma sencilla de programar la lectura de un dato aislado. En cambio, un fichero de acceso aleatorio es lo que su nombre sugiere: un fichero del que se puede leer un campo de cualquier registro sin tener que recorrer el fichero entero. En la figura 7.2 se ilustra la diferencia entre los dos métodos de archivo. Los sistemas de ficheros de acceso aleatorio requieren unidades de disco; no obstante, podemos diseñar programas que producen más o menos el mismo efecto utilizando fichero secuenciales. Estudiemos, pues, este tipo de ficheros.

CREACIÓN DE UN FICHERO

Como siempre que abordamos algo nuevo, vamos a empezar por lo más sencillo. Veamos en primer lugar cómo establecer la conexión entre el ordenador y el microdrive. El ejemplo de la figura 7.3 ilustra cómo se puede grabar un dato, concretamente del valor de la variable A, en el microdrive número 1. En la línea 10 *abrimos un fichero*. La instrucción es la palabra OPEN_NEW, seguida del número de canal (del #3 al #15), el número del microdrive y el nombre del fichero. En la línea 10 de este programa la instrucción es OPEN_NEW #7,MDV1_PRUEBA; su efecto es abrir un fichero nuevo en el microdrive 1, a través del canal #7, con el nombre PRUEBA. Si se trata de abrir con la instrucción OPEN_NEW un fichero que ya exista en la cinta, el ordenador emite un mensaje de error. Esto constituye una protección contra el borrado accidental de ficheros preexistentes. Análogamente, si utiliza la instrucción OPEN_IN con un cartucho en el que no haya ningún fichero grabado, se produce un mensaje de error.

El siguiente paso es asignar un valor a la variable A (línea 20). La línea 30 es la fundamental. La instrucción de esta línea significa «transmitir por el canal #7 el valor de A». Puesto que el canal #7 ha sido asignado al micro-

```
10 OPEN_NEW #7,MVD1_PRUEBA
20 A=5
30 PRINT#7,A
40 CLOSE #7
```

Fig. 7.3 Grabación del valor de una sola variable.

drive en la línea 10, el efecto de la línea 30 es «escribir» en la cinta. El valor queda archivado con el nombre de «PRUEBA», de forma que podrá ser encontrado fácilmente cuando se lo busque por ese nombre. En realidad, la línea 30 no provoca inmediatamente la grabación física. El sistema de grabación envía al microdrive los caracteres agrupados; el sistema operativo continúa agrupando caracteres hasta que se llena cierta zona de la memoria, denominada *tampón*. Cuando se abre un canal, el sistema operativo automáticamente reserva una zona de memoria para utilizarla como tampón. Al ejecutar una línea tal como la 30, los datos que han de ser grabados son transferidos al tampón. Si éste se llena, los datos que contiene son enviados al microdrive; esto puede ocurrir varias veces en el curso de una grabación. Si el tampón no llega a llenarse, los datos no son grabados mientras no se cierre el canal; al cerrar el canal, el ordenador vacía el tampón y envía los datos acumulados al microdrive. Esta acción la realiza la instrucción CLOSE # 7 (línea 40).

```
5 CLS
10 OPEN_IN #5,MVD1_PRUEBA
20 INPUT#5,A
30 PRINT A
40 CLOSE #5
```

Fig. 7.4 Lectura de un dato grabado en microdrive.

Por complicadas que sean estas transferencias de datos, al usuario lo único que le interesa es que el microdrive gira durante un tiempo bastante largo y luego se para. El cursor vuelve a aparecer mucho antes de que el microdrive se detenga. Esto ocurre porque el dato ha sido transferido al tampón, y a partir de ese momento el ordenador queda a la espera de nuevas instrucciones, mientras que el microdrive hace su trabajo independientemente. Ahora tenemos que comprobar que el dato ha quedado grabado correctamente. Para ello escribimos el programa de la figura 7.4. En la línea 10 el fichero se abre con OPEN_IN, no con OPEN_NEW, pues lo que queremos es abrir un fichero que ya existe, no crear uno nuevo. El número de canal es ahora el # 5, elegido caprichosamente: además, tenemos que especificar el número del microdrive y el nombre del fichero. Recuerde que el signo de subrayar (_) es *ineludible* cuando se deba especificar el nombre de un fichero después del número del microdrive.

Una vez abierto el fichero, podemos leer los datos. De ello se encarga la línea 20, INPUT # 5,A. Si hubiéramos puesto INPUT a secas, estaríamos haciendo referencia al teclado; al especificar # 5, obligamos al ordenador a leer

en el dispositivo asignado al canal 5. La línea 30 escribe el dato leído y la línea 40 cierra el canal. Todo esto parece, y es, muy sencillo. Pero debemos observar estos dos programas muy atentamente porque contienen la esencia de lo que vamos a necesitar en nuestro trabajo de archivo.

La figura 7.5 contiene un ejemplo un poco más ambicioso, en el que generamos un fichero de números (los pares del 0 al 50) al tiempo que los grabamos en microdrive y los escribimos en la pantalla. La línea 10, con su instrucción OPEN_NEW, conecta un canal a un dispositivo. El canal es el número 5, aunque podríamos haber elegido cualquier número del 3 al 15. El dispositivo es el microdrive número 1 (MDV1) y el nombre del fichero es «PARES». Observe que cuando se conecta un canal con un microdrive, es necesario especificar el número del microdrive y el nombre del fichero.

```
10 OPEN_NEW #5,MDV1_PARES
20 CLS:FOR N=0 TO 50 STEP 2
30 PRINT #5,N
35 PRINT N;" ";
40 NEXT N
50 CLOSE #5
```

Fig. 7.5 Grabación de un fichero de números.

Resumiendo, la línea 10 conecta #5 con el fichero «PARES» en el microdrive 1. La ejecución de esta línea hace que el microdrive se ponga en marcha para buscar un lugar de la cinta en el que se pueda grabar el nombre del fichero y los datos. La siguiente etapa es la generación de los datos, mediante el bucle que empieza en la línea 20. Este bucle asigna a la variable N los números pares comprendidos entre 0 y 50. ¿Cómo enviamos los números al canal? De ello se encarga la línea 30, con la instrucción PRINT #5,N.

Pero ya sabemos que no basta con enviar los datos al canal #5 para que queden grabados en el cartucho. El microdrive es más rápido que el cassette, pero muchísimo más lento que el propio ordenador. La ejecución del bucle FOR ... NEXT se completa antes de que el motor del microdrive llegue ni siquiera a ponerse en marcha. Cada vez que se ejecuta la línea 30, se envía un dato al tampón. Si el número de datos fuera mayor, la grabación empezaría cuando se llenara el tampón; pero en este caso el bucle se completa antes de que eso ocurra. La grabación empieza cuando se ejecuta la línea 50. La instrucción CLOSE #5 vacía el tampón correspondiente al canal 5. Como este canal está asignado al microdrive, la acción de la línea 50 es poner en marcha el microdrive, grabar en la cinta el nombre del fichero, vaciar el tampón, grabar los datos y cerrar el canal. Si se ejecuta el programa, se puede oir

mo el microdrive busca un tramo de cinta libre y graba los datos. No se aprecia en qué momento se produce la grabación de los datos, ya que la línea 35 escribe los números en la pantalla a medida que la línea 30 los envía al tampón.

Se puede considerar un tampón como si fuera un amortiguador, un depósito de agua o una cuenta bancaria; cualquier mecanismo que hace posible que las entradas y salidas puedan efectuarse en momentos distintos. El QL crea un tampón cada vez que se abre un canal; la capacidad del tampón es muy grande, como se puede comprobar con el programa de la figura 7.6.

```
10 OPEN_NEW #6,MDV1_MASPARES
20 CLS:FOR N=0 TO 5000 STEP 2
30 PRINT #6,N
40 PRINT N
50 NEXT N
60 CLOSE #6
```

Fig. 7.6 Un fichero de números mucho más largo con cuya grabación se puede observar el funcionamiento del tampón y del microdrive.

Esta vez los datos ocupan varios millares de bytes. Cuando ejecute el programa observará que el microdrive gira continuamente hasta algún tiempo después de que los números hayan terminado de salir en la pantalla. El tampón está siendo llenado por la línea 30 y vaciado hacia el microdrive. Como el tampón es muy grande, no llega a llenarse completamente. Cuando termina el bucle pueden quedar datos en el tampón; la línea 60 los envía al microdrive y cierra el canal.

De lo dicho se puede deducir que la instrucción CLOSE es muy importante. Sin ella, los ficheros pequeños no se graban en absoluto, y los grandes se graban incompletos. Otro efecto de la instrucción CLOSE es grabar una señal de *fin de fichero*. Esta señal permite al ordenador identificar el final del fichero cuando lo lea más tarde; sin ella el microdrive no sabría dónde terminar la lectura, e incluso podría empezar a leer el fichero siguiente. El efecto de omitir la instrucción CLOSE es lo que se llama en informática «dejar los ficheros abiertos».

Estos programas han grabado datos en el cartucho del microdrive, pero al lector puede quedarle la duda de si efectivamente algo ha quedado grabado en la cinta. Vamos a estudiar el método que permite leer de un fichero, pero veamos antes cómo se puede comprobar los datos grabados de una forma muy sencilla. Teclee esta orden directa:

COPY MDV1_PARES TO SCR (copiar ... en la pantalla)

Cuando pulse ENTER oirá el movimiento del microdrive, y poco después aparecerán en la pantalla los números pares. La pantalla se desplazará hacia arriba a medida que se vayan escribiendo los números, de forma que cuando el microdrive se detenga ya sólo serán visibles los últimos. También se pueden comprobar los datos de un fichero largo por este procedimiento. Teclee

COPY MDV1_MASPARES TO SCR

y pulse ENTER. Observará que el microdrive se mueve durante un corto tiempo, entrega datos al tampón y éste los envía a la pantalla. En este caso el cuello de botella es la pantalla, pues la visualización en la pantalla es más lenta que la lectura de datos en el microdrive.

Observemos detenidamente estas potentes órdenes. Hemos utilizado COPY para conectar un dispositivo con otro. El dispositivo «fuente» es un fichero del microdrive 1; el dispositivo destino es SRC, o sea, la pantalla. El efecto de la orden es «copiar» el fichero entero de la cinta a la pantalla. La utilidad de esta orden se basa en que permite inspeccionar un fichero sin tener que cargar un programa que lo abra y lo lea. El inconveniente es que obliga a recorrer el fichero completo, de principio a fin. Para examinar solamente una parte de un fichero se puede interrumpir el proceso pulsando CTRL y la barra espaciadora en el instante en que acabe de aparecer en la pantalla el texto que se quería consultar.

En la figura 7.7 se da otro método de comprobación de los datos grabados en un fichero. En este ejemplo podremos observar nuevamente el funcionamiento del tampón. La línea 10 abre el fichero «MASPARES». En lugar de con COPY, esta vez leemos los datos con instrucciones INPUT a través del canal #7. Si nos hubiéramos limitado a programar un bucle para leer los datos y enviarlos a la pantalla, el efecto de este programa sería muy similar al de COPY, pues el microdrive estaría continuamente llenando el tampón, y la pantalla vaciándolo. Pero en el programa hemos incluido interrupciones para hacer una pausa cada vez que se llena la pantalla (líneas 50 a 80). La

```
10 OPEN_IN #7,MDV1_MASPARES
20 FOR N=0 TO 5000 STEP 2
30 INPUT #7,N
40 PRINT N
50 IF N/40=INT(N/40) THEN
60 FOR J=1 TO 1000:NEXT J
70 CLS
80 END IF
90 NEXT N
```

Fig. 7.7 Lectura de datos grabados en microdrive. Cada vez que se llena la pantalla se hace una pausa.

condición de la línea 50 es IF N/40 = INT(N/40), que se cumple cuando N es múltiplo de 40, pues es éste el único caso en el que la división no tiene resto y el cociente N/40 es igual a su parte entera. Cada vez que N es múltiplo de 40, la línea 60 introduce un retardo, al final del cual se borra la pantalla y el programa continúa. Al ejecutar este programa se puede oir cómo el microdrive se pone en marcha y se detiene de vez en cuando. Mientras el microdrive está funcionando, está enviando datos al tampón; en cambio, durante la pausa de la línea 60 no se ejecuta la instrucción INPUT #7, por lo que el microdrive se para a esperar. Se puede detener el programa pulsando simultáneamente CTRL y la barra espaciadora, y reanudarlo tecleando CONTINUE (ENTER).

OTROS FICHEROS SECUENCIALES

Supongamos que lo que se desea grabar no es un conjunto de números generado por un programa, sino un conjunto de nombres introducidos por el teclado. Para el microdrive esto no representará nada nuevo, pues lo que graba es conjuntos de datos, sin importarle su naturaleza. Cada vez que se pulsa ENTER en una instrucción INPUT, los datos son enviados al tampón, y permanecen en él hasta que se llena o hasta que el proceso de captación termina y se cierra el canal. Nuevamente podemos apreciar aquí la necesidad de disponer de un tampón: no sería lógico que el motor del microdrive tuviera que ponerese en marcha cada vez que pulsáramos una tecla.

En la figura 7.8 se da un programa de este tipo. Normalmente, para recopilar información de este tipo habría que almacenar los nombres asignándo-

```
10 OPEN_NEW #6,MDV1_NOMBRES
20 CLS:AT 16,2:PRINT"NOMBRES."
30 PRINT"Este programa graba nombres en el"\"mic
rodrive. Teclee los nombres a"\"medida que se lo
s vaya pidiendo."\" (X para terminar)."
40 REPeat INTRNOMB
50 INPUT"El nombre es: ";nm$
60 IF nm$="X" OR nm$="x" THEN
70 EXIT INTRNOMB
80 END IF
90 PRINT #6,nm$
100 END REPeat INTRNOMB
110 CLOSE#6
```

Fig. 7.8 Grabación de un conjunto de nombres en microdrive.

los a una lista, lo que obligaría a dimensionarla previamente. Pero esto no es necesario, a menos que se quiera consultar o modificar algún elemento introducido con anterioridad. Las cosas no serán tan fáciles cuando leamos el fichero, pero de eso ya hablaremos más tarde.

Analicemos el programa. La línea 10 abre el canal y el fichero como de costumbre. El canal es en este caso el #6, el dispositivo es el microdrive número 1 y el fichero es «NOMBRES». Cuando se ejecuta el programa, esta línea hará que el microdrive busque un trozo libre de cinta, grabe en él el nombre del fichero y quede preparado para grabar los datos. Las líneas 20 y 30 dan instrucciones al usuario. El bucle REPEAT empieza en la línea 40; su función es captar y enviar al tampón nombres hasta que la cadena recibida sea «X» o «x». Para que el programa sea útil, en la práctica habrá que dar al usuario instrucciones más detalladas. Los nombres se pueden introducir de cualquier forma, pues el QL, a diferencia de otros ordenadores, admite el uso de la coma; por ejemplo GARCIA, MANUEL.

La línea 60 comprueba si el nombre introducido es «x». Si no es así, la línea 90 envía el nombre nm$ al tampón. Si los nombres se introducen deprisa y de forma continua, el microdrive funcionará continuamente atendiendo al vaciado del tampón; si se hace una pausa, el microdrive se detendrá, y puede no volver a ponerse en marcha hasta el momento en que se ejecute la línea 110. Una vez grabados los datos en el microdrive, se puede hacer una copia de seguridad de la siguiente forma: ponga el cartucho que contiene los datos en MDV1, un cartucho inicializado en MDV2 y ejecute la orden

COPY MDV1_NOMBRES TO MDV2_

LECTURA DE FICHEROS

En este momento el lector tendrá grabados varios ficheros en un cartucho, tanto de números como de nombres. Vamos a dedicar mayor atención a los métodos de lectura y a la posterior utilización de la información leída. Supongamos, por ejemplo, que los números que hemos grabado en el fichero PARES hubieran sido generados por un programa de contabilidad. Una cosa que podríamos necesitar sería leer los datos del fichero para sumarlos. Ésta es una función habitual en informática. La figura 7.9 indica cómo realizarla. El programa empieza por borrar la pantalla; la línea 20 abre el canal #7 y lo conecta con el fichero MDV1_PARES. Recordemos que en este fichero habíamos grabado los números pares del 0 al 50, es decir, un total de 26 números. Para leerlos todos, el bucle FOR ... NEXT de la línea 40 tiene que ser ejecutado 26 veces. La variable «Total» ha sido puesta a cero en la línea 30. La línea 50 lee un número del fichero y lo asigna a la variable J.

```
10 CLS
20 OPEN_IN #7,MDV1_PARES
30 Total=0
40 FOR N=1 TO 26
50 INPUT#7,J
60 Total=Total+J
70 NEXT N
80 PRINT"La suma es ";Total
90 CLOSE #7
```

Fig. 7.9 Lectura de un fichero de números para sumarlos.

La línea 60 suma cada nuevo dato al total. Al final del programa, la línea 80 escribe en la pantalla la suma; la línea 90 cierra el fichero.

¿Qué ocurriría si no supiésemos cuántos datos contiene el fichero? No podríamos utilizar un bucle FOR ... NEXT, pues no sabríamos qué valores límite ponerle. La solución es sencilla si sabemos cuál es el último dato grabado en el fichero. Sabemos que en este caso es el número 50, por lo que podemos leer datos hasta encontrarnos con el 50. Esto es lo que hace el pro-

```
10 CLS
20 OPEN_IN #8,MDV1_PARES
30 Total=0
40 REPeat entrdato
50 INPUT #8,J
60 Total=Total+J
70 IF J=50 THEN
80 EXIT entrdato
90 END IF
100 END REPeat entrdato
110 PRINT"La suma es ";Total
```

Fig. 7.10 Detección del último elemento de un fichero.

grama de la figura 7.10. Las líneas 70 a 90 son las fundamentales. Mientras el dato leído sea distinto de 50, el programa volverá a la línea 50 a leer y sumar otro dato. Cuando se lee el número 50, se ejecuta primero la línea 80 y luego la 110.

¿Qué ocurre si nos olvidamos de programar la detección del último dato del fichero? Modifique la línea 70 de la siguiente forma:

70 IF J = 70 THEN

Al ejecutar el programa el microdrive funciona como siempre, pero el ordenador escribe el mensaje

At line 50 end of file (fin de fichero en la línea 50)

en la última línea de la pantalla, y en la zona superior no escribe nada. Los datos han sido leídos y sumados, como se puede comprobar con la orden GOTO 110. El mensaje de error indica que se han agotado los datos y que no se puede seguir ejecutando la línea 50. Es muy desagradable que un programa se detenga con un mensaje de error; sería muy conveniente que este ordenador dispusiese de una instrucción BASIC que recondujera el programa a la línea 110 (en este ejemplo) en caso de error. Muchos sistemas basados en disco permiten detectar la señal de fin de fichero (EOF) y obrar en consecuencia cuando ésta aparezca. Mi QL acepta la línea 70 IF EOF THEN, pero esta instrucción no parece tener efecto. La otra posibilidad, 70 IF J = EOF, hace que la máquina quede bloqueada y tenga que ser reinicializada con el botón RESET.

Así pues, no nos queda otro remedio que grabar por nuestra cuenta un indicador de «fin de fichero», aparte de la señal que el microdrive graba automáticamente. El indicador de «fin de fichero» hace posible leer datos hasta que el dato leído sea precisamente ese indicador, momento en el que cerraremos el fichero para evitar el mensaje de error «end of file». En el último ejemplo sabíamos que el último dato grabado era el número 50, y por eso pudimos utilizarlo como indicador de fin de fichero. En cada aplicación se puede elegir un dato distinto. Por ejemplo, si se está creando un fichero de números y se sabe que nunca se va a teclear el 999, se puede grabar este número al final de los datos captados por el teclado. Otro número posible sería el 0, siempre que se sepa que ningún dato tiene ese valor.

Otra posibilidad es grabar CHR$(0) al final de cada fichero. Este carácter debe ser leído como cadena literal, no como número. La dificultad consiste ahora en reconocer una cadena cuando se está leyendo un fichero de números. Afortunadamente, los números se pueden leer *como si fueran* cadenas literales. La figura 7.11 muestra un programa en el que se genera un conjunto de números y se los graba en un fichero llamado «FINCHR». Los datos son números aleatorios comprendidos entre 10 y 100. Al final del bucle que genera y archiva los datos se graba CHR$(0) (línea 50). La línea 70 detiene el programa para que usted tenga tiempo para pensar en qué va a consistir la siguiente etapa. En la cinta hemos grabado unos cuantos números en forma de códigos ASCII, no en la forma en la que el QL almacena internamente las variables numéricas. Por consiguiente, nada nos impide leer los números *como si fueran caracteres literales*. Si usted pone J en lugar de J$ en este programa, seguramente le ocurrirá, como a mí, que el QL se bloquea y responde a las órdenes LIST y NEW con el mensaje «bad line», con la consiguiente pérdida del programa.

```
10 CLS:OPEN_NEW #6,MDV1_FINCHR
20 FOR N=1 TO 10
30 J=RND(10,100)
40 PRINT #6,J:NEXT N
50 PRINT #6,CHR$(0)
60 CLOSE #6
70 STOP
100 CLS:OPEN_IN #7,MDV1_FINCHR
110 REPeat leer
120 INPUT #7,J$
130 IF J$=CHR$(0) THEN
140 EXIT leer
150 END IF
160 PRINT J$
170 END REPeat leer
180 CLOSE #7
```

Fig. 7.11 Utilización del carácter CHR$(0) como «indicador de fin de fichero».

Para pasar a la segunda parte del programa teclee CONTINUE (ENTER). El microdrive lee los números en forma de cadena, J$. El programa realiza la comprobación de la línea 130 antes de escribir cada número en la pantalla. Cuando encuentra CHR$(0), el programa salta a la línea 180 y cierra el fichero. Mientras el carácter leído no sea CHR$(0), el programa retorna a 120 para leer la siguiente cadena. A pesar de que los números se leen en forma de cadena literal, se los puede utilizar como números (por ejemplo, para realizar con ellos operaciones aritméticas).

LECTURA DE FICHEROS DE NOMBRES

Ya hemos aprendido a grabar y leer un fichero de números; es hora de que aprendamos los diveros métodos que podemos aplicar a la lectura del fichero de nombres que creamos antes. Recordemos que al crear ese fichero fuimos grabando los nombres uno tras otro y que no pusimos ningún indicador de fin de fichero al final. Esto hace que el fichero sea más difícil de leer, a menos que sepamos cuántos nombres contiene. Desde luego, siempre se puede copiar el fichero a la pantalla (con la orden COPY). La figura 7.12 da otro método, en el que se leen los caracteres uno a uno mediante INKEY$. La línea 20 escribe el carácter leído; el signo de punto y coma impide el salto a la línea siguiente. También podríamos asignar los caracteres a una lista en lugar de escribirlos. Cuando grabamos el fichero, el ordenador puso auto-

```
10 CLS:OPEN_IN #6,MDV1_nombres
15 REPeat leenombr
20 PRINT INKEY$(#6);
30 END REPeat leenombr
```

Fig. 7.12 Lectura de un fichero con INKEY$.

```
10 OPEN_NEW #6,MDV1_OTROSNOM
20 CLS:AT 16,2:PRINT"NOMBRES."
30 PRINT\"Este programa graba nombres en el"\"mi
crodrive. Teclee los nombres a"\"medida que se l
os vaya pidiendo."\" (X para terminar)."
40 REPeat captar
50 INPUT"El nombre es: ";N$
60 IF N$="X" OR N$="x" THEN
70 EXIT captar
80 END IF
90 PRINT #6,N$
100 END REPeat captar
110 PRINT #6,CHR$(0)
120 CLOSE #6
150 STOP
200 CLS:AT 16,2:PRINT"LECTURA DEL FICHERO"
210 OPEN_IN #7,MDV1_OTROSNOM
220 REPeat leer
230 INPUT #7,J$
240 IF CHR$(0) THEN
250 EXIT leer
260 END IF
270 PRINT J$
280 END REPeat leer
290 CLOSE #7
```

Fig. 7.13 Programa más refinado para la captación, grabación y reproducción de un conjunto de nombres. Se utiliza CHR$(0) como indicador de fin de fichero.

máticamente el retorno del carro (código 13) entre cada dos nombres sucesivos. En la lectura, ese mismo código hace que los nombres aparezcan en la pantalla en líneas distintas. Este proceso de lectura producirá el mensaje de error «At line 20 end of file», con el que se detendrá el programa.

Como en el caso de los ficheros numéricos, la solución está en grabar un número conocido de datos con un bucle FOR ... NEXT, o bien un «indica-

dor de fin de fichero» adecuado al final de cada fichero. Un indicador posible es CHR$(0). En la figura 7.13 se muestra un programa de grabación y lectura de nombres que graba CHR$(0) como indicador de fin de fichero. Con este programa se puede grabar un número arbitrario de datos; el proceso de lectura se detiene automáticamente al final del fichero. Las líneas 200 a 290 leen el fichero; la línea 240 detecta la presencia de CHR$(0). Mientras no aparezca CHR$(0), el programa volverá a la línea 230 a leer otro nombre. Cuando aparece CHR$(0), el programa cierra el fichero.

Este método sirve para leer datos de un fichero y escribirlos en la pantalla, pero para la misma función sirven también INKEY$ y COPY. Lo que normalmente se necesita es asignar los datos a una lista que el ordenador pueda manipular más tarde. Las listas permiten, por ejemplo, ordenar alfabéticamente los nombres. Pero no todas las aplicaciones requieren la utilización de listas. Supongamos, por ejemplo, que queremos buscar los nombres que empiezan por la letra «J». El programa de la figura 7.14 realiza esta función.

Las dos primeras líneas ya nos son conocidas. Al ejecutar el programa, la línea 10 llena el tampón, de manera que si los datos del fichero no son muchos, todos ellos estarán en el tampón antes de que el programa pase a la línea siguiente. La línea 30 pide al usuario que teclee la inicial del nombre (no olvide la distinción entre mayúsculas y minúsculas). Si se introduce más de una letra, la línea 40 selecciona la primera. En la línea 50 empieza un bucle en el cual en cada pasada se lee un nombre del fichero y se comprueba si el dato leído es el indicador de fin de fichero. Si no lo es, la línea 100 compara la inicial antes introducida con la primera letra del nombre. Si se ha

```
10 CLS:OPEN_IN #9,MDV1_OTROSNOM
20 AT 14,2:PRINT"BUSCADOR DE NOMBRES"
30 AT 2,4:INPUT"La inicial es: ";Q$
40 Q$=Q$(1)
50 REPeat buscar
60 INPUT #9,N$
70 IF N$=CHR$(0) THEN
80 PRINT"Fin del fichero;"\"no hay ningun nombre
 con esa inicial"
90 EXIT buscar
100 ELSE IF Q$=N$(1) THEN
110 PRINT"El nombre es ";N$
120 EXIT buscar
130 END IF
140 END REPeat buscar
150 CLOSE #9
```

Fig. 7.14 Búsqueda de nombres en un fichero.

leído el fichero completo y no se ha encontrado ningún nombre que empiece por esa letra, la línea 70 termina el programa. En cambio, si se encuentra un nombre que cumpla la condición, la línea 110 lo escribe y la búsqueda termina.

Este programa funciona bien con pequeñas cantidades de datos, pero no tanto si éstos llenan el tampón varias veces. El proceso es lento si hay que cargar los datos del microdrive uno a uno. Si el número de datos es muy grande, no hay alternativa. Si no, lo usual en este tipo de programas es asignar los datos a una lista para tenerlos todos ellos en la memoria del ordenador. Entonces el microdrive se utiliza solamente como «almacén», pero todas las operaciones de selección se realizan en la memoria del ordenador. Se preguntará el lector por qué toda esta complicación, con lo fácil que sería poner los datos en líneas de DATA en el programa. La razón es que los datos pueden ser generados por un programa y utilizados por otros programas distintos. El programa que lee y utiliza los datos puede ser así bastante corto, lo que permite que en la memoria quepan muchos datos. Dicho de otra forma, el sistema de microdrive facilita la utilización de programas cortos que operen con muchos datos.

En la figura 7.15 se muestra cómo pasar datos de un fichero grabado en microdrive a una lista. Por supuesto, podríamos haber empezado por grabar una lista mediante una instrucción de la forma

PRINT #9,A$(J)

pero mi intención en este capítulo es analizar el sistema de microdrive, no revisar las técnicas que el lector ya debiera conocer. Cuando nos ponemos a leer nombres para asignarlos a una lista nos encontramos con un problema: tenemos que dimensionar la lista de forma que pueda contener todos los nombres. Para ello necesitamos saber de antemano cuántos nombres vamos a leer y cuántos caracteres tiene el nombre más largo. Si no tuvimos la precaución de grabar estos datos al principio del fichero, ¿qué podemos hacer?

```
10 CLS:OPEN_IN #9,MDV1_OTROSNOM
20 AT 14,2:PRINT"BUSCADOR DE NOMBRES."
30 AT 2,4:PRINT"Escriba la inicial del nombre."\
"(0 para terminar)."
40 J=1:L=0:nu=0:tomadato:DIM N$(nu,L)
50 REPeat entrada
60 INPUT #9,A$
70 IF A$=CHR$(0) THEN
80 EXIT entrada
90 END IF
100 N$(J)=A$:J=J+1
110 END REPeat comparar
```

```
120 Z=J-1
130 REPeat comparar
140 REPeat pruebalt
150 INPUT "Primera letra: ";Q$
160 Q$=Q$(1)
170 IF Q$="O" THEN
180 EXIT comparar
190 END IF
200 M=0
210 FOR X=1 TO Z
220 IF N$(X)(1)=Q$ THEN
230 PRINT N$(X)
240 M=1
250 NEXT X
260 ELSE NEXT X
270 END IF
280 IF M=0 THEN
290 PRINT"No encuentro ningun nombre con esa"\"i
nicial. Pruebe con otra."
300 END IF
310 END REPeat pruebalt
320 END REPeat comparar
330 CLOSE #9:CLEAR
340 PRINT"FIN DEL PROGRAMA"
350 STOP
360 DEFine PROCedure tomadato
370 REPeat finfich
380 INPUT #9,A$
390 IF A$=CHR$(0) THEN
400 EXIT finfich
410 END IF
420 CH=LEN(A$)
430 IF CH>L THEN
440 L=CH
450 END IF
460 nu=nu+1
470 END REPeat finfich
480 CLOSE #9
490 OPEN_IN #9,MDV1_OTROSNOM
500 END DEFine
```

Fig. 7.15 Buscador de nombres más avanzado en el que se asignan los datos a los elementos de una lista. Para dimensionar la lista correctamente se leen todos los nombres y se determina la longitud del más largo.

La respuesta es muy sencilla. Leemos todos los nombres, los contamos y medimos su longitud; guardamos estos dos datos en un par de variables. Esta operación se puede programar en un procedimiento. A continuación utilizamos esos datos para dimensionar la lista. El único defecto de este método es que nos obliga a repasar toda la cinta (desde que terminamos de leer el fichero para contar los datos hasta que lo volvemos a leer para asignar los datos a la lista). El tiempo que se tarda no es mucho, dada la rapidez del microdrive, y además es casi independiente del número de datos. En cualquier caso, es un precio que hay que pagar para poder dimensionar la lista correctamente, es decir, para no provocar un error al intentar leer más datos que los previstos y para no truncar nombres de longitud mayor que la prevista. Si no tenemos que sobredimensionar la lista, estaremos aprovechando al máximo la memoria del QL, y de esto puede depender el que todos los datos quepan en la memoria o no.

En la figura 7.15 se da el listado de este programa. La novedad del programa está en la línea 40 y en el procedimiento «tomadato» que se define en la línea 360 y siguientes. En la línea 40 se inicializan las variables numéricas nu (número de elementos) y L (longitud). El procedimiento «tomadato» asigna los valores correctos a estas variables, antes de que las utilicemos en la instrucción DIM de la línea 40. El procedimiento consiste en un bucle que lee todos los nombres, uno por uno, hasta que detecta la presencia de CHR$(0). Las líneas 430 a 450 actualizan el valor de L (longitud del nombre más largo); se compara la longitud del último nombre leído, CH, con la máxima longitud hasta el momento, L; si el último nombre leído es el más largo hasta ahora, se hace L = CH. Cuando se hayan leído todos los nombres, L será la longitud del más largo. La variable «nu» es el contador de nombres leídos, sin contar el indicador de fin de fichero. En este programa tenemos que ser más cuidadosos que en casos anteriores, pues pretendemos dimensionar la lista con precisión. Así, en la línea 120 hacemos Z igual a J-1, ya que el contador J ha sido incrementado *antes* de detectar el indicador de fin de fichero.

Observemos que tenemos que abrir y cerrar el fichero *dos veces*: una vez para contar y medir los nombres; como en esta operación llegamos al final del fichero, tenemos que cerrarlo (ya que no podemos hacer retroceder la cinta hasta el principio) antes de poder volver a leer datos de él. Si omitiéramos las instrucciones CLOSE y OPEN de las líneas 480 y 490, provocaríamos el error «end of file» en la línea 60. Tenga el lector en cuenta este detalle, pues es algo que puede no ocurrírsele cuando esté diseñando un programa similar.

Volvamos al programa, línea 40. Una vez ejecutado el procedimiento «tomadato», ya podemos dimensionar la lista, pues ya sabemos cuántos nombres vamos a leer y cuál es la longitud máxima que necesitamos. Ahora tenemos que volver a leer los nombres del fichero para asignarlos a los elementos

de la lista. Para ello hemos programado el bucle «entrada». Su funcionamiento es muy sencillo: la línea 60 lee una cadena del fichero, comprueba si se trata de CHR$(0) y, si no lo es, la asigna a un nuevo elemento de la lista (línea 100). Al mismo tiempo se incrementa el contador J. Al final del bucle «entrada» el valor de J excede en una unidad al número de elementos contenidos en la lista; la línea 120 corrige el valor de J.

En la línea 130 empieza el bucle principal, en el que se trabaja con la lista. Este bucle se extiende hasta la línea 330, pero se lo puede interrumpir tecleando un «0» (cero, no letra «O»). La acción principal se realiza en el bucle «pruebalt»; en la línea 150 se capta la inicial. Como antes, se toma solamente la primera letra de la respuesta tecleada por el usuario. La línea 100 permite salir del bucle «comparar» si la cadena introducida es «0». Observe el lector que esto implica salir también del bucle «pruebalt». Hay pocas versiones de BASIC que permitan hacer algo similar. El riesgo es que la memoria quede corrupta; en previsión de ello se puede incluir una instrucción CLEAR al final del programa. Más adelante hablaremos de este asunto. En la línea 200 se pone a cero la «variable indicadora» M. Una variable de este tipo es el mecanismo que se utiliza para seguirle la pista a la situación de cierto acontecimiento; en este caso, M indica si se ha encontrado un nombre que cumpla la condición especificada; más adelante veremos para qué sirve esto. En la línea 210 empieza un bucle FOR ... NEXT que examina todos los nombres de la lista. Si se encuentra un nombre que empieze por la letra especificada, las líneas 220 a 240 se encargan de escribirlo en la pantalla y de poner a 1 la variable M. La línea 250 es NEXT X porque, aunque se haya encontrado un nombre, debemos seguir examinando la lista por si hay otro que satisfaga la condición. De no hacerlo así, el programa sólo encontraría el *primer* nombre de la lista que cumpliera la condición. Si el nombre examinado no empieza por la letra especificada, la línea 260 pasa al siguiente elemento de la lista.

Veamos cómo funciona la variable indicadora M. Si después de examinar todos los nombres la variable M sigue siendo 0, esto indica que ninguno de los nombres empieza por la letra tecleada por el usuario. Las líneas 280 a 300 examinan el valor de M y actúan en consecuencia. A continuación llegamos al final del bucle «pruebalt». Como el usuario puede querer probar con otra letra, en la línea 320 ponemos la instrucción END REPEAT comparar.

MODIFICACIÓN DE LOS FICHEROS

Dejemos claro desde el principio que no es posible alterar el contenido de los ficheros grabados en microdrive. Lo que sí se puede hacer es leer el fichero, modificarlo y volver a grabarlo. Puesto que el QL tiene dos unidades de

microdrive, se puede grabar la versión actualizada del fichero, *con el mismo nombre*, en el segundo microdrive. Es muy importante poder actualizar un fichero sin tener que cambiarle el nombre. El sistema de microdrive está diseñado de forma que es imposible borrar un fichero grabando sobre él otros datos con el mismo nombre. La figura 7.16 indica cómo utilizar los dos microdrives para actualizar un fichero.

No obstante, hay riesgos que el manual del QL no menciona. Cuando probé este programa descubrí que el ordenador «corrompía» los datos; para evitarlo tuve que poner instrucciones CLEAR en diversos lugares del programa. Mientras no se subsanen estos defectos, recomiendo al lector que no confíe datos valiosos al sistema de microdrice sin antes haber comprobado exhaustivamente el programa.

```
10 CLS:AT 13,6:PRINT"POR FAVOR, ESPERE."
20 CLEAR
30 OPEN_IN #5,MDV1_OTROSNOM
40 OPEN_NEW #6,MDV2_OTROSNOM
50 REPeat copialis
60 INPUT #5,A$
70 IF A$=CHR$(0) THEN
80 EXIT copialis
90 END IF
100 PRINT #6,A$
110 END REPeat copialis
120 CLOSE #5:CLEAR
130 CLS:AT 13,2:PRINT"AMPLIACION DEL FICHERO"
140 PRINT"Escriba los nombres nuevos con los"\"q
ue quiera completar el fichero."\"(X para termin
ar.)"
150 REPeat nuevaent
160 INPUT"Introduzca un nombre: ";A$
170 IF A$="X" OR A$="x" THEN
180 EXIT nuevaent
190 END IF
200 PRINT #6,A$
210 END REPeat nuevaent
220 PRINT #6,CHR$(0)
230 CLOSE #6
240 CLEAR
250 DELETE MDV1_OTROSNOM
```

Fig. 7.16 Actualización de un fichero utilizando los dos microdrives. Se graba la versión actualizada en el segundo microdrive y luego se borra el fichero original.

El programa empieza escribiendo en la pantalla el mensaje «POR FAVOR, ESPERE» para que el usuario no se impaciente durante el medio minuto que tardan los microdrives en copiar, del 1 al 2, el fichero OTROSNOM. La instrucción CLEAR de la línea 20 evita la mezcla de signos «x» y «£» con los nombres. La línea 30 abre MDV1 para leer; la línea 40 abre MDV2 para escribir el nuevo fichero. El bucle «copialis» lee una cadena de MDV1, la compara con CHR$(0) y, si no es igual a este carácter, la graba en MDV2. Cuando la cadena leída es CHR$(0), el bucle termina *sin grabar en* MDV2 *el indicador de fin de fichero*. Se cierra el canal de lectura y se ejecuta otra instrucción CLEAR para evitar la corrupción del siguiente conjunto de datos.

La introducción de datos nuevos empieza en la línea 130; en ella y en la siguiente se dan instrucciones al usuario. A continuación se entra en el bucle «nuevaent»; la instrucción INPUT capta un nombre para grabarlo en MDV2. No se oirá el movimiento del microdrive mientras no se llene el tampón. Si se omite la instrucción CLEAR de 120, el ordenador introduce caracteres espurios en los nombres y emite un mensaje que dice que ocurre algo raro en la línea 210. El proceso de introducción de nombres nuevos se interrumpe tecleando «X» o «x», lo que provoca la salida del bucle hacia la línea 220. Ésta graba el indicador de fin de fichero, CHR$(0). La línea 230 cierra el canal de salida. En la línea 240 hemos puesto CLEAR para que nos traiga suerte. Finalmente, la línea 250 borra (DELETE = borrar) el fichero original (que está en MDV1). Es buena idea omitir esta línea mientras no se haya comprobado que el QL no ha hecho de la suyas con la versión actualizada del fichero.

Dejemos claro por qué se debe borrar el fichero original. Una vez actualizado el fichero, el cartucho en el que se ha grabado la versión actualizada pasa a ser el nuevo archivo original. Para la siguiente ejecución del programa ya habrá que insertarlo en MDV1, mientras que el cartucho que contiene la primera versión tendrá que ir al MDV2. Si no se borrara el primer fichero, en la segunda actualización no se podría realizar la copia de ficheros, pues ello representaría escribir en un fichero ya existente. Quizá piense el lector que estoy insistiendo demasiado en esto; sé por experiencia que éste es el error que se comete con mayor frecuencia cuando se empieza a manejar los microdrives, sobre todo cuando antes se ha trabajado con unidades de disco en otras máquinas. Muchos sistemas de disco permiten escribir un fichero sobre otro preexistente con el mismo nombre. No ocurre lo mismo con el sistema de microdrive del QL, en el que el borrado accidental de ficheros es más difícil.

8

Ventanas y otros efectos

Trabajar con un ordenador nunca es aburrido, pero hay recursos con los que se puede hacer el trabajo mucho más interesante. Uno de los efectos especiales que ofrecen los ordenadores modernos es la posibilidad de utilizar ventanas. En este capítulo vamos a examinar ese efecto y algunos otros que permiten organizar la pantalla de forma mucho más atractiva que la mera visualización de texto y columnas de números. Las ventanas son un buen punto de partida, pues es muy probable que el lector no haya experimentado con ellas, a no ser que antes haya tenido un ordenador capaz de manejarlas.

Una *ventana* no es más que una región de la pantalla que puede ser utilizada independientemente de la pantalla en su conjunto. En una pantalla se puede escribir texto, se la puede desplazar por *scrolling* o se la puede borrar, con independencia de lo que esté ocurriendo en el resto de la pantalla. El manejo de ventanas se complica en el QL por el hecho de que son varias las instrucciones que permiten definirlas, y también por lo difícil que es «cerrar» una ventana después de definirla. Veamos, pues, cómo controlarlas. La figura 8.1 da una idea de los efectos que se pueden conseguir. La primera acción del programa es escribir un listado. La idea es disponer de un texto suficientemente amplio como para demostrar la forma de las ventanas. A continuación se crea una ventana en la parte superior de la pantalla; la anchura es casi la de la pantalla completa; observe la ausencia del borde que normalmente tiene la pantalla principal. Después de una pausa se crea una ventana en la zona inferior de la pantalla. Esta ventana, lo mismo que la anterior, tiene fondo rojo; observe que la escritura se adapta a la anchura de la ventana igual que si se tratara de la pantalla completa. Observe también cómo se desplaza el texto hacia arriba en la ventana. A continuación se define una ventana que ocupa toda la pantalla. Cuando se escribe texto en esta ventana se pueden perder caracteres en ambos extremos, ya que el televisor no visualiza la pantalla completa. Finalmente se restaura el borde y se escribe otro mensaje.

```
10 LIST
20 WINDOW 471,25,20,20:CLS
30 PRINT"Esta es la ventana superior"
40 espera
50 WINDOW 200,30,156,200:CLS
60 PRINT"Esta es la ventana inferior"
70 espera
80 PRINT"Se desplaza como la pantalla completa"
90 espera
100 PRINT"y se borra con CLS"
110 espera
120 CLS:espera
130 WINDOW 511,255,0,0:CLS
140 PRINT"Esta es la pantalla completa, sin bord
e"
150 espera
160 BORDER 20,0
170 PRINT:PRINT:PRINT"Ahora ya tiene borde"
200 DEFine PROCedure espera
210 LOCal N
220 FOR N=1 TO 1500
230 NEXT N
240 END DEFine
```

Fig. 8.1 Ventanas en la pantalla del QL.

Este programa habrá dado al lector una idea de lo que se puede conseguir manejando adecuadamente las ventanas. Vamos a analizar el programa para aprender los conceptos en los que se basa la definición de ventanas. El primer concepto es el de «coordenada»; aparece por primera vez en la línea 20.

Un sistema de coordenadas es una forma de especificar posiciones mediante números, algo semejante a los números que se utilizan en la instrucción AT. El «origen» es el punto de referencia a partir del cual se miden las distancias. En el QL, el origen para la instrucción WINDOW y algunas otras está en el *extremo superior izquierdo* de la pantalla. Para los usuarios que utilicen un televisor este punto estará fuera de la pantalla y será, por consiguiente, invisible. Las unidades de distancia son tales que, cualquiera que sea el tamaño del televisor, la pantalla tiene una anchura de 512 unidades y una altura de 256 unidades (Fig. 8.2). La posición de un punto en horizontal se especifica mediante la coordenada X; en vertical, mediante la coordenada Y. La coordenada X es la distancia del punto en cuestión al borde izquierdo de la pantalla; la coordenada Y es la distancia del punto al borde superior de la pantalla. Por lo tanto, el punto de coordenadas X = 255,

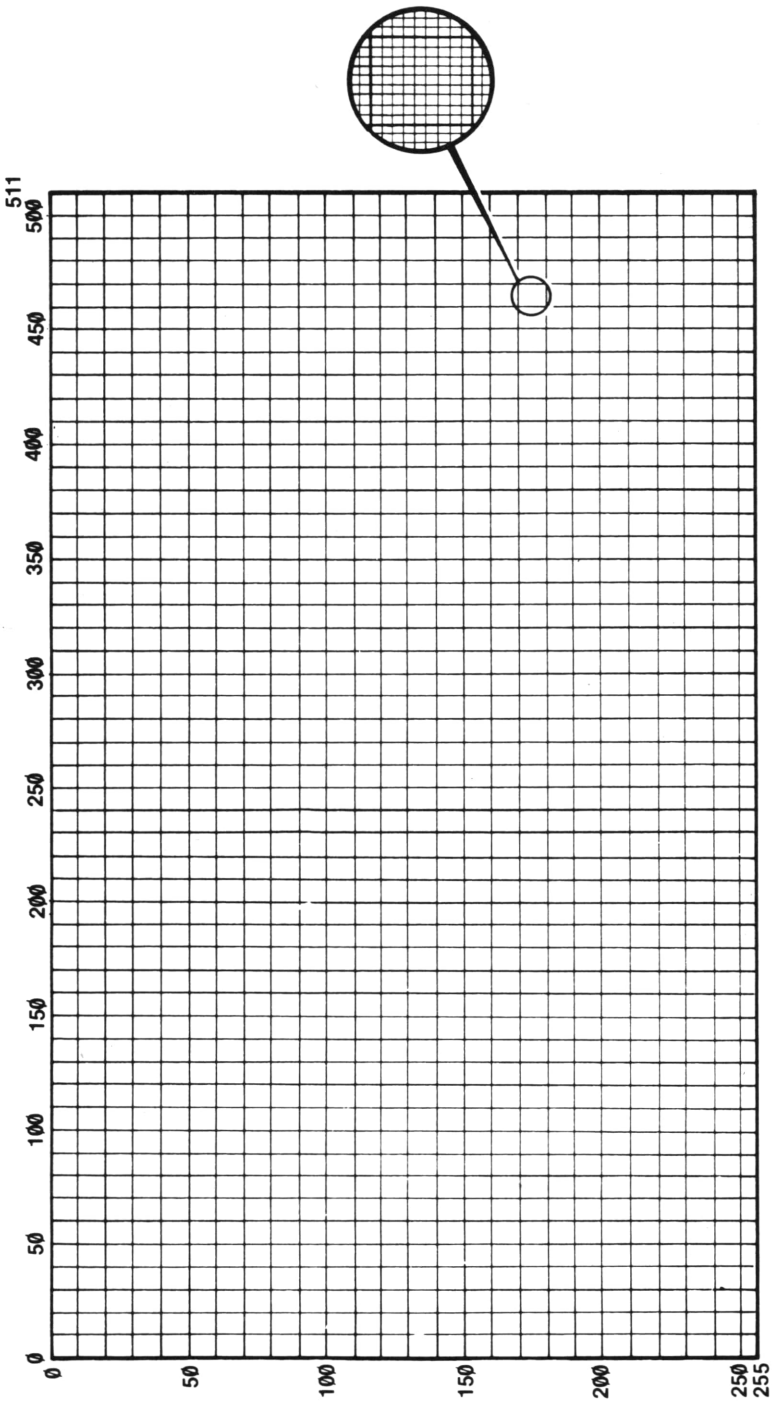

Fig. 8.2 Sistemas de coordenadas para las instrucciones de este capítulo (distinto del sistema que se utiliza en los gráficos).

Y = 127 estará en el centro de la pantalla. Las coordenadas no varían de 1 a 512 y de 1 a 126, respectivamente, sino de 0 a 511 y de 0 a 255. Esto, como ya sabemos, es lo más frecuente en informática.

Volviendo al programa, línea 20, observemos que la instrucción WINDOW va seguida de cuatro números (parámetros), separados por comas. Los dos primeros números especifican el *tamaño* de la ventana (el primero es la anchura y el segundo es la altura). En este caso, la ventana tiene 471 unidades de anchura y 25 unidades de altura. En 25 unidades en vertical caben dos líneas de texto. Los dos últimos números especifican la posición del origen de la ventana. En este caso hemos elegido 20 y 20, de forma que el extremo superior izquierdo de la ventana está muy cerca del de la pantalla. La instrucción CLS afecta solamente a la ventana, no al resto de la pantalla. Análogamente, toda instrucción PRINT que se ejecute después de definir una ventana afectará solamente a esa ventana.

¿Cómo podemos cambiar esta situación? Una forma es definir otra ventana, que es lo que hace la línea 50. Esta vez la ventana es más estrecha: 200 unidades. Como el valor máximo de X es 512, 200 es aproximadamente 2/5 de la anchura de la pantalla. La altura es solamente 25, y el origen está en X = 156, Y = 200. Este valor de X hace que la ventana quede centrada en anchura. Esto se debe a que 512 − 200 = 312, y 312/2 = 156. La distribución en horizontal es, pues, la siguiente: 156 unidades libres, 200 para la ventana y otras 156 libres a la derecha. Con Y = 200 ponemos el origen relativamente cerca del extremo inferior de la pantalla. Las siguientes líneas de programa escriben texto en esta ventana; este texto se adapta a la anchura de la ventana. La instrucción CLS borra solamente la ventana. La primera ventana que definimos, lo mismo que el resto de la pantalla, no es afectada por esta instrucción.

Si seguimos así, habremos dividido la pantalla en porciones independientes entre sí. Para volver a la situación normal ejecutamos la línea 130. En ella definimos una ventana que cubre toda la pantalla, pues hemos especificado la anchura y altura máximas posibles y hemos puesto otra vez el origen en 0, 0. Lo que ocurre ahora es que no toda la ventana es utilizable, en particular con un televisor. Seguramente se perderán algunas de las letras escritas por la línea 140 cerca del borde izquierdo de la pantalla. Esto ocurre porque hemos eliminado el borde de seguridad que el ordenador establece automáticamente al encenderlo. La línea 160 restaura ese borde. Los parámetros de la instrucción BORDER son: el primero, el tamaño del borde, 20 unidades; el segundo, el color, negro. Más adelante hablaremos de los colores del QL. Si se omite este parámetro, el color será igual que el del resto de la pantalla. De la línea 200 en adelante se define el procedimiento «espera», mediante el cual introducimos una pausa en diversos puntos del programa. También podríamos haber utilizado la instrucción PAUSE.

VENTANAS ASIGNADAS A CANALES

La instrucción WINDOW es muy útil, y todavía tenemos que aprovecharla mejor. Pero no podemos estar definiendo ventanas para luego perder el control sobre ellas en cuanto definamos otras. Hay otra forma de definir ventanas, asignándolas a alguno de los canales de los que ya hemos hablado. De esta forma, siempre podemos actuar sobre una ventana determinada sin más que especificar el número de canal correspondiente, aun cuando hayamos definido y utilizado otras. La clave está en la instrucción OPEN, seguida de SCR para especificar la pantalla.

```
10 CLS
20 OPEN#4,SRC_200X30A156X20
30 LIST #4
40 PAUSE 200
50 OPEN#5,SCR_400X50A56X200
60 PAUSE 200
70 CLS #4
80 PAUSE 200
90 PRINT #4,"Texto en la ventana superior"
100 PAUSE 200
110 CLS#5:PRINT #5,"Texto en la ventana inferior
"
120 PAUSE 200
130 PRINT#5,\\\\\
140 CLS#4:PRINT #4,"ventana inferior desplazada"
150 PAUSE 200
160 CLS
170 PRINT"Se borra la pantalla completa"
180 CLOSE#4:CLOSE#5
```

Fig. 8.3 Conexión de una ventana con un canal.

Veamos el ejemplo de la figura 8.3. Se borra la pantalla completa y a continuación se abre el canal #4 conectándolo a la pantalla:

OPEN #4,SCR_200X30A156X20

Esto significa, evidentemente, que el canal #4 se conecta a la pantalla (SCR). 200X30 especifica el tamaño de esta pantalla (en realidad, de una ventana). Esta «X» es un separador que no podemos omitir ni confundir con la coordenada X. La «A» es otro separador; los dos últimos números son, como es de suponer, la posición del origen de la ventana. Una vez ejecutada

esta instrucción, ya podemos referirnos a esa ventana en instrucciones del tipo PRINT #4 o CLS #4. Mientras no cerremos este canal o lo asignemos a algún otro dispositivo, seguirá controlando esa ventana. Las instrucciones PRINT y CLS seguirán afectando a la pantalla completa. Las ventanas y la pantalla completa se solapan mutuamente; es decir, si borramos la pantalla se borran también las ventanas.

Para probar la existencia de la ventana definida en la línea 20, la siguiente línea lista el programa en ella. La instrucción LIST sola, sin #4, afectaría a la pantalla principal. Después del listado introducimos una pausa y asignamos otra ventana al canal #5. El resto del programa demuestra cómo se manejan las ventanas independientemente la una de la otra. Obsérvese cómo se desplaza hacia arriba la ventana inferior escribiendo en ella una sucesión de caracteres de «siguiente línea». Nótese también que la instrución CLS de 160 borra la pantalla entera, incluidas las ventanas.

DESPLAZAMIENTO EN VERTICAL DE LA PANTALLA

Hasta ahora el lector habrá observado el desplazamiento en vertical (*scrolling*) de la pantalla al hacer listados o al escribir textos suficientemente largos en ella. No obstante, el desplazamiento de la pantalla se puede controlar directamente mediante la instrucción SCROLL. La figura 8.4 muestra los

```
10 CLS
20 WINDOW 75,255,218,0
30 PRINT"PRUEBA"
40 FOR N=1 TO 240
50 SCROLL 1
60 NEXT N
70 FOR N=240 TO 1 STEP -1
80 SCROLL -1
90 NEXT N
```

Fig. 8.4 La instrucción SCROLL.

efectos que se consiguen, difíciles de lograr de otra forma. En este ejemplo la palabra PRUEBA se desplaza lentamente hacia abajo, por efecto del bucle que contiene la instrucción SCROLL 1. El parámetro de SCROLL especifica en cuántas unidades se debe desplazar la pantalla. En este caso el desplazamiento se realiza en 240 etapas, lo que produce un movimiento muy suave. Si el parámetro es un número negativo, el desplazamiento se realiza

en sentido inverso, de forma que la palabra se mueve hacia arriba. Para hacer más lento (aunque menos suave) el movimiento se puede intercalar una pausa en cada pasada del bucle FOR ... NEXT.

El efecto de la instrucción SCROLL se puede hacer más complejo. Un segundo parámetro controla el tipo de desplazamiento. Veamos el programa de la figura 8.5. La primera parte es muy parecida al programa anterior; se

```
10 CLS
20 WINDOW 200,255,156,0
30 escribir
40 FOR N=1 TO 150
50 SCROLL 1,1
60 PAUSE 2
70 NEXT N
80 FOR N=1 TO 140
90 SCROLL -1
100 NEXT N
110 CLS
120 CURSOR 0,20
130 escribir
140 CURSOR 0,20
150 FOR N=1 TO 250
160 SCROLL 1,2
170 NEXT N
180 STOP
190 DEFine PROCedure escribir
200 PRINT"Esta"\"es"\"una"\"frase"\"de"\"prueba"
\"de"\"la"\"instruccion"\"SCROLL"\"del"\"QL."
210 PRINT"Asi"\"se"\"muestra"\"su"\"efecto."
220 END DEFine
```

Fig. 8.5 Desplazamiento selectivo de la pantalla poniendo un segundo parámetro en SCROLL.

borra la pantalla y se define una ventana. La línea 30 escribe un texto mediante un procedimiento. El bucle que empieza en la línea 40 produce un desplazamiento, pero la instrucción que lo realiza es ahora SCROLL 1,1. El segundo «1» significa que el desplazamiento no afecta a las líneas que están por debajo del cursor. Ahora bien, el procedimiento «escribir» ha dejado el cursor sobre la última palabra, de forma que el bucle desplaza las líneas anteriores, pero no la última. Hemos ralentizado la acción con una instrucción PAUSE. Terminado el primer bucle, el segundo desplaza la frase completa; ahora SCROLL sólo lleva un parámetro.

Para el siguiente efecto de magia utilizamos SCROLL 1,2. Esta instrucción desplaza el texto que está *por debajo* del cursor. Esto no tendría nada de interesante si el cursor estuviera en la última línea del texto, de modo que lo hemos colocado al principio con la instrucción CURSOR 0,20 (línea 120). Los parámetros de esta instrucción tienen el significado usual de coordenadas X, Y, pero referidos al origen de la ventana; por ejemplo, si se especifica CURSOR 0,0, el cursor se sitúa en el origen de la ventana, no en el de la pantalla. Así pues, CURSOR 0,20 pone el cursor en el borde izquierdo de la ventana, 20 unidades por debajo de su extremo superior. El procedimiento «escribir» empieza a escribir en ese punto. La línea 140 vuelve a situar el cursor en la primera línea; SCROLL 1,2 desplaza las líneas que están por debajo de ésta. Nada nos ha obligado a poner el cursor en la primera línea; el lector puede experimentar con otras posiciones. Incluso es posible situarlo a media altura de una palabra, con lo que los efectos posibles son ilimitados. Por ejemplo, ponga CURSOR 0,76 en la línea 140.

Hay una instrucción similar, PAN, que desplaza la pantalla en horizontal. La forma de utilizarla es muy parecida y no es necesario que demos un ejemplo. Además, el manual es suficientemente claro en este caso.

ESCRITURA MÁS ATRACTIVA

Vamos a explorar las posibilidades de hacer más atractiva la visualización de textos. Para ello empezaremos con los modificadores de la instrucción PRINT. Como el nombre indica, estos modificadores controlan la forma de escribir en la pantalla. La acción es independiente del tipo de caracteres escritos, ya sean letras, dígitos o formas gráficas. En la figura 8.6 se da un ejemplo, aplicado a una de estas instrucciones de «efectos especiales»: FLASH (parpadear). FLASH se «enciende» cuando el número que le sigue es 1, y se apaga cuando el número es 0. Así pues, el efecto de esta instrucción se aprecia en todo lo que se escriba entre FLASH 1 y FLASH 0 y permanece

```
10 CLS
20 PRINT"Esta es una frase normal"
30 PAUSE 150
40 FLASH 1
50 PRINT"Esta parpadea"
60 PAUSE 100
70 FLASH 0
80 PRINT"Esta no"
```

Fig. 8.6 Programación de textos parpadeantes.

hasta que se hagan desaparecer los caracteres afectados, bien sea por desplazamiento de la pantalla, por la acción de CLS o por sustituir los caracteres por otros. FLASH sirve, evidentemente, para atraer la atención del usuario hacia algún lugar de la pantalla.

Otra instrucción de este tipo es OVER. Los parámetros de esta instrucción pueden ser 0, 1 o −1. La figura 8.7 da un ejemplo de OVER 1. Con el parámetro 1, OVER activa la sobreimpresión de caracteres; es decir, las letras

```
10 CLS
20 AT 2,4:PRINT"E T   S L   N T U C O   V R D L
Q "
30 PAUSE 200
40 OVER 1
50 AT 2,4:PRINT" S A E   A I S R C I N O E   E
 L"
```

Fig. 8.7 La instrucción OVER.

```
10 CLS
20 AT 4,4:PRINT"Lycee Superieur"
30 AT 4,6:PRINT"El lenguaje de programacion"
40 OVER 1
50 AT 7,4:PRINT"´"
60 AT 27,6:PRINT"´"
70 AT 10,8:PRINT"± 2% de tolerancia"
80 AT 10,8:PRINT"_"
90 OVER 0
```

Fig. 8.8 Utilización de la instrucción OVER para acentuar palabras y crear signos especiales.

nuevas se escriben sobre las anteriores, pero no las reemplazan. El efecto no siempre es agradable, pero en ocasiones puede ser útil. En el programa de la figura 8.7 se escribe la mitad de las letras de un título y luego, tras un pausa, se escribe el resto. Otra aplicación de OVER 1 es la creación de caracteres con acentos, tildes y diéresis para escribir en castellano, francés, etc. En la figura 8.8 se da un ejemplo. OVER 1 permite poner acentos agudos en las letras «e» y «o». Otro signo que se puede construir de esta forma es el «más menos» (±) (líneas 70 y 80). En este caso hay que poner el signo de subrayar porque el signo − no está a la altura correcta. El resultado es aceptable; para mejorarlo se podría escribir el signo − ajustando previamente la posición del cursor con CURSOR.

```
10 CLS:OVER 0
20 A$="OOOOOVVVVVXXXXX"
30 B$="IIIIIIIIIIIIIII"
40 AT 2,4:PRINT A$
50 PAUSE 150
60 OVER 1
70 AT 2,4:PRINT B$
80 PRINT"CON OVER 1"
90 PAUSE 100
100 OVER 0
110 AT 2,8:PRINT A$
120 PAUSE 150
130 OVER -1
140 AT 2,8:PRINT B$
150 PRINT"CON OVER -1"
160 OVER 0
```

Fig. 8.9 Diferencias entre OVER 1 y OVER −1.

En la figura 8.9 se ilustra la diferencia entre OVER 1 y OVER −1. A$ es un conjunto de letras y B$ es una sucesión de letras «I» que vamos a superponer a A$. La primera vez utilizamos OVER 1 y la segunda, OVER −1. Observe el lector que las letras que se escriben con OVER −1 tienen un color diferente y que la sobreimpresión produce formas muy distintas, al menos en la pantalla del televisor. OVER 0 vuelve el modo de escritura a la situación normal. El modo de sobreimpresión se mantiene hasta que se ejecuta OVER 0 o hasta que se apaga y enciende el ordenador.

Probemos ahora el programa de la figura 8.10. Primero se escribe una frase y luego, tras una pausa, se ejecuta STRIP 0 antes de OVER 0. El efecto es el normal de OVER 0 (las letras nuevas reemplazan a las anteriores), pero el fondo es de diferente color. El color es el especificado por el parámetro de STRIP.

```
10 CLS
20 AT 2,4:PRINT"OTRA PRUEBA CON OVER 0"
30 PAUSE 100
40 STRIP 0
50 OVER 0
60 AT 2,4:PRINT"IIIIIIIIIII"
```

Fig. 8.10 Utilización de STRIP para resaltar textos.

```
10 CLS:UNDER 1
20 AT 16,2:PRINT"TITULO"
30 UNDER 0
40 AT 2,4:PRINT"Utilizacion de ";:UNDER 1:PRINT"
under 1";
50 UNDER 0:PRINT"para subrayar"
```

Fig. 8.11 UNDER 1 subraya textos.

La instrucción UNDER tiene dos variantes. UNDER 1 subraya todos los textos que se envían a la pantalla. Este modo se cancela con UNDER 0. Estas instrucciones se incluyen normalmente entre dos instrucciones PRINT sucesivas, generalmente terminadas en signos de punto y coma. La figura 8.11 da un ejemplo en el que el lector puede basar sus experimentos.

CÓMO ESCRIBIR EN COLOR

Vamos a empezar a hablar de las instrucciones que controlan los colores en el QL. Las más importantes son BORDER, PAPER e INK. Empecemos por BORDER; ya hemos encontrando esta instrucción cuando hablamos de la definición de ventanas. Como sabemos, el primer parámetro de BORDER especifica el tamaño del borde; el segundo especifica el color. También se puede incluir un número de canal si se quiere hacer referencia al borde de una ventana. Cuando se está utilizando la pantalla normal, el borde especificado por la instrucción será *interior a la zona del borde normal*. Si se crea un borde, las instrucciónes PRINT enviarán el texto a la zona de la pantalla

```
10 CLS
20 FOR n=0 TO 7
30 BORDER 5*n,n
40 PAUSE 100
50 NEXT n
60 FOR n=7 TO 0 STEP -1
70 BORDER 5*n,n
80 PAUSE 100
90 NEXT n
100 AT 7,5
110 PRINT"Exhibicion de colores"
```

Fig. 8.12 Creación de bordes de diferentes colores.

que es interior a ese borde. Lo que sí se puede hacer es definir primero un borde grueso y luego otro fino; entonces sí es posible escribir en el primer borde. La definición de bordes requiere cierto cuidado, como demuestra el ejemplo de la figura 8.12. En este programa hay dos bucles que generan dos conjuntos de bordes de tamaños y colores diferentes. A medida que crece *n*, los bordes se van superponiendo a los anteriores; el efecto es el de un borde que se va haciendo cada vez más grueso y va cambiando de color. En el segundo bucle los bordes van siendo cada vez más finos; cada uno deja una franja del anterior, con lo que al final se tiene un borde multicolor. Si se escribe algo en este momento, hay que utilizar la instrucción AT para enviar el texto a la «pantalla», es decir, a la zona interior al borde más grueso.

Número	Color
0	Negro
1	Azul
2	Rojo
3	Magenta (rojo + azul)
4	Verde
5	Cyan (azul + verde)
6	Amarillo
7	Blanco

Fig. 8.13 Los números de los colores. Estos números se utilizan como parámetros en BORDER y en otras instrucciones que controlan el color.

En la figura 8.13 se da la lista de colores de la instrucción BORDER. Los números van del 0 al 7. Si se utilizan números mayores que éstos (en un televisor), los colores parecen tener superpuesta una especie de rejilla. Los números superiores a 128 producen efectos extraños; no se los debe utilizar con un televisor. Cuando yo los probé al televisor no le ocurrió nada, pero algunos números producían un desagradable efecto «moiré» y otros generaban rayas verticales en la pantalla.

Las siguientes instrucciones son PAPER (papel) e INK (tinta). Su efecto se deduce de su nombre: PAPER especifica el color del fondo; INK, el de los caracteres o signos que se envían a la pantalla. El margen de colores es también de 0 a 7. En la figura 8.14 se ilustra la forma de utilizar estas instrucciones. La ejecución de PAPER no hace aparecer por sí misma un color en la pantalla; es decir, afecta solamente al fondo sobre el que se escriba en

```
10 CLS
20 FOR N=0 TO 7
30 PAPER N
40 CLS
50 AT 10,12
60 PRINT"PAPER = ";N
70 PAUSE 100
80 NEXT N
90 PAPER 0:CLS
100 FOR N=0 TO 7
110 INK N
120 PRINT"INK = ";N
130 PAUSE 100
140 NEXT N
```

Fig. 8.14 Demostración de las instrucciones PAPER e INK.

el futuro. Para que la pantalla entera resulte afectada hay que ejecutar CLS (línea 40 del programa). En la segunda parte del programa el fondo es negro (PAPER 0) y se escribe con diferentes colores de tinta. Este ejemplo prueba que, para que los textos sean legibles, los colores de PAPER e INK tienen que estar bien contrastados, lo que se consigue con números lo más distintos posible. Mi preferencia es texto claro sobre fondo oscuro, pues el inverso, texto oscuro sobre fondo claro, da una imagen inestable y muy molesta.

No es de esperar que los colores de las letras sean muy buenos en un televisor, pues los televisores de color no están diseñados para esta tarea específica. Si además tenemos en cuenta que el 90 por 100 de los varones no discrimina perfectamente los colores, se comprende que los mejores efectos de color se consigan solamente en las condiciones más favorables: con un monitor y utilizando colores fuertes en superficies grandes.

Si se utilizan números altos (por ejemplo, 255) en la instrucción INK, se obtienen resultados *muy* extraños, al menos en el televisor.

Cada vez que se asigna un color a PAPER, automáticamente se asigna el mismo color a STRIP. Así pues, si se desea ejecutar una instrucción OVER 0 después de PAPER, es necesario redefinir STRIP.

OTRAS INSTRUCCIONES

Para cada color se pueden especificar hasta tres números, pero solamente si se utiliza un monitor. En el momento de escribir estas líneas todavía no he podido probar el QL con un monitor de color, por lo que no sé cómo serán estos colores.

Hay diversas instrucciones que tienen interés en la preparación de la pantalla. Una de ellas es CLS seguida de un número, que permite elegir qué parte de la pantalla debe ser borrada. El significado de los parámetros es análogo al que tienen en la instrucción SCROLL; el manual los describe detalladamente.

BLOCK es una instrucción especializada que se utiliza en la programación de diagramas de barras. También esta instrucción está bien descrita en el manual y no requiere más explicación. SCALE es una potente instrucción que permite variar el tamaño de las imágenes; la estudiaremos en el capítulo siguiente. Ya hemos visto que CSIZE cambia el tamaño de los caracteres.

9

Introducción a los gráficos

La capacidad gráfica del QL es muy superior a la de los ordenadores de la serie ZX, Spectrum incluido. El término *gráficos* hace referencia a los dibujos y formas que se pueden dibujar en la pantalla; las instrucciones que los manejan son muy distintas de las que hemos utilizado para los textos. Un concepto que tenemos que definir es el de *resolución*. La resolución de los gráficos es el número de puntos que necesitamos para llenar la pantalla y podemos controlar individualmente. Si el lector observa de cerca la pantalla de un televisor de color verá que está dividida en líneas de puntos. Estos puntos son los que emiten la luz, de modo que ninguna imagen puede ser menor que la anchura de un punto o que la distancia de una línea a otra. Los «puntos» con los que trabaja un ordenador son los llamados *pixels* (de *picture elements*, elementos de imagen). Cada pixel corresponde en tamaño a varios puntos de un televisor. El QL ofrece dos resoluciones distintas. En baja resolución controla 256 pixels en horizontal y otros tantos en vertical, es decir, un total de 65536 pixels, cada uno de los cuales puede tener un color de los ocho disponibles (incluido el blanco y el negro). Muchos fabricantes llamarían a esto «alta resolución»; de hecho, es la máxima aprovechable es un televisor de color. En alta resolución el QL controla 512 pixels en horizontal y 256 en vertical, es decir, 131072 pixels en total. Tan gran resolución tiene un precio, que es la limitación del número de colores: solamente negro, rojo y verde. Por otra parte, esta resolución sólo se puede apreciar si se utiliza un monitor.

El cambio de una resolución a otra se hace con la instrucción MODE. Si al encender el QL se ha pulsado la tecla F2, el ordenador estará en modo de «baja resolución». Para cambiarlo se teclea MODE 4 o MODE 512. En el momento de pulsar ENTER se borra la pantalla. Los caracteres que se escriban a continuación serán de tamaño mucho menor. Con MODE 8 o MODE 256 se vuelve a baja resolución.

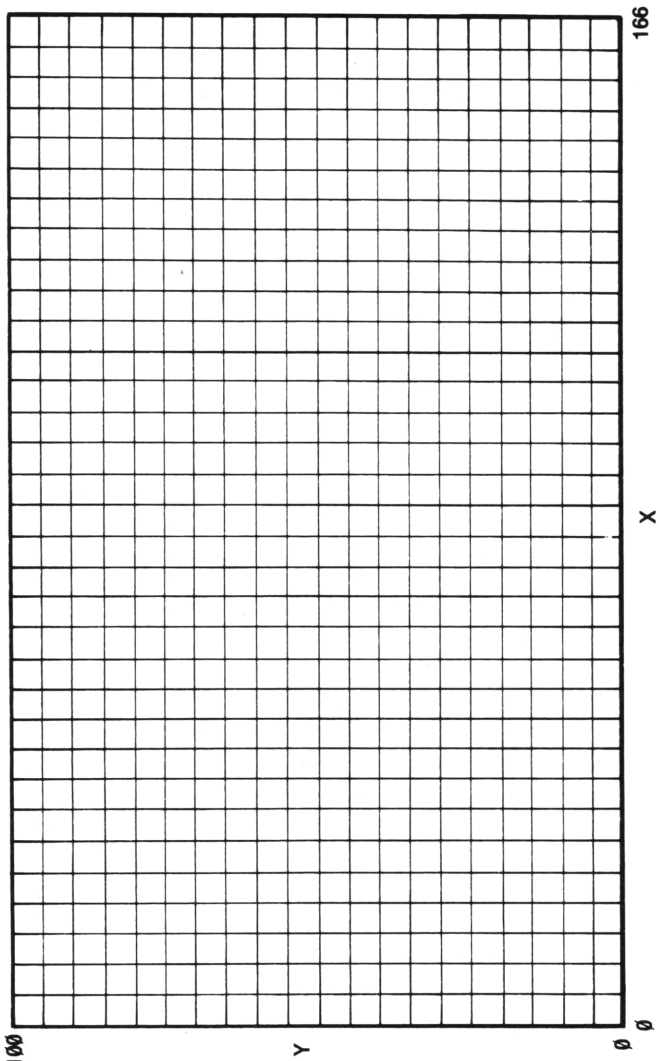

Fig. 9.1 Sistema de coordenadas implícito para la pantalla gráfica. Los márgenes de variación de X e Y se pueden cambiar con la instrucción SCALE. Obsérvese que el origen está en el extremo inferior izquierdo de la pantalla.

DIBUJO DE CIRCUNFERENCIAS

La posibilidad de «pintar» cada uno de los 65536 puntos (o pixels) no sería útil en absoluto si para crear imágenes hubiera que especificar la posición y el color de cada punto uno por uno. Hay ordenadores así de torpes, pero

el QL, como el lector ya habrá supuesto, es bastante más hábil. De hecho, hay gran variedad de instrucciones gráficas que permiten dibujar de forma más eficaz. Todas estas instrucciones utilizan un sistema de coordenadas de X, Y similar al que ya conocemos. La diferencia está en el margen de valores de las coordenadas. Si no se cambia la escala (con la instrucción SCALE), el margen de Y es de 0 a 100, y el de X aproximadamente de 0 a 166 (Fig. 9.1). Este último número depende del ajuste del televisor. Lo importante es que las unidades de distancia son iguales en los dos ejes, independientemente de la resolución con que se trabaje. De esta forma, los programas gráficos funcionan en cualquier grado de resolución sin más que cambiar el modo. La otra diferencia entre este sistema de coordenadas y el del capítulo anterior es la diferente posición del origen. En este sistema el origen está en el extremo inferior izquierdo. Las distancias verticales se miden *hacia arriba*; las horizontales, *hacia la derecha*.

```
10 MODE 256:CLS
20 FOR N=5 TO 50 STEP 5
30 INK RND(0,7)
40 CIRCLE 83,50,N
50 PAUSE 100
60 NEXT N
```

Fig. 9.2 Dibujo de una circunferencia con la instrucción CIRCLE.

En la figura 9.2 se muestra un programa que dibuja circunferencias. La línea 10 restablece el modo de baja resolución, por si se lo hubiera cambiado. No hemos cambiado el color de PAPER. En la línea 20 empieza un bucle; los valores de la variable del contador serán los radios de las circunferencias. La línea 30 selecciona al azar los colores de INK; esto significa que algunas circunferencias se dibujarán con el color del fondo y serán, por lo tanto, invisibles. La línea 40 dibuja las circunferencias. La instrucción CIRCLE va seguida de tres parámetros. Los dos primeros son las coordenadas del centro de la circunferencia. Como queremos centrarla en la pantalla, hemos puesto 83 y 50. El tercer parámetro es el radio. Hemos puesto como radio la variable N, de forma que el bucle dibuja un conjunto de circunferencias de diámetro creciente.

La siguiente instrucción que vamos a estudiar es FILL (rellenar). El efecto de FILL es rellenar la figura cerrada que se va a dibujar a continuación. La instrucción tiene que ser ejecutada antes que la que realice el dibujo. Su forma correcta es FILL 1; si no se especifica un color de tinta, el ordenador rellenará la figura con el último color de tinta utilizado. La figura 9.3 da un

```
10 MODE 256:CLS
20 INK 6:FILL 1
30 CIRCLE 83,50,30
40 PAUSE 100
50 INK 3:FILL 1
60 CIRCLE 83,50,45
70 INK 6:FILL 0
80 CIRCLE 83,50,20
```

Fig. 9.3 Rellenado de un recinto cerrado.

ejemplo. Las líneas 10 a 30 dibujan una circunferencia y la rellenan con tinta amarilla. Cada vez que se quiera rellenar un recinto hay que programar FILL 1, si bien se puede cancelar la función de rellenado con FILL 0, como demuestran las líneas 50 a 80. Tal como aparecen en el listado, dibujan una circunferencia azul dentro de una amarilla. Si se omite la instrucción FILL 0 de la línea 70, lo que se rellena es la mitad del círculo interior y una porción de la corona circular. Para que la circunferencia amarilla no se rellene es necesario el FILL 0 de la línea 70. El manual no deja esto claro.

MODALIDADES DE CIRCLE

La instrucción CIRCLE es capaz de hacer bastante más que una simple circunferencia. Con una sola instrucción se puede dibujar un conjunto de circunferencias; los grupos de parámetros van separados por signos de punto y coma. En la figura 9.4 se da un ejemplo muy deportivo. Esta instrucción

```
10 MODE 256:INK 6
20 CIRCLE 22,30,20;52,40,20;82,30,20;112,40,20;1
42,30,20
```

Fig. 9.4 Dibujo de varias circunferencias con una sola instrucción CIRCLE.

permite también dibujar elipses; las elipses tienen la forma que se muestra en la figura 9.5, en la que se indica también cómo dibujarlas en un papel con un hilo, un par de alfileres y un lapicero. Las elipses tienen centro, pero no un radio definido. Para caracterizarlas se utilizan los ejes (diámetros) mayor y menor.

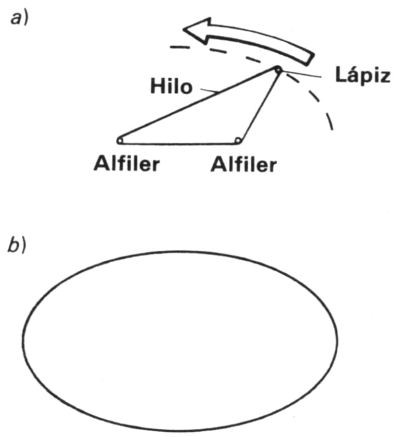

Fig. 9.5 Dibujo manual con dos alfileres, un hilo y un lápiz (*a*). Si se mantiene tenso el hilo, el resultado es una elipse (*b*).

Además hay que especificar si el eje mayor es horizontal o vertical o, en general, qué ángulo forma con el eje vertical. Todo esto se especifica con solo *dos* números: el primero sirve para «deformar» una circunferencia, indicando la relación entre el eje mayor y el menor (es decir, la *excentricidad* de la figura); el segundo es el ángulo que forma el eje mayor con la vertical.

Veamos el ejemplo de la figura 9.6. En todas las elipses que dibuja este programa el ángulo es cero radianes. En el primer bucle la excentricidad varía de 0,1 a 1: las primeras elipses son muy estrechas y se van ensanchando a medida que crece E; la excentricidad 1 corresponde a una circunferencia. En el segundo bucle la excentricidad crece de 1 a 2, con lo que las elipses resultan aún más anchas.

```
10 MODE 256:INK 4
20 FOR E=.1 TO 1 STEP .1
30 CIRCLE 83,50,30,E,0
40 PAUSE 50
50 NEXT E
60 PAUSE 150:INK 5
70 FOR E=1 TO 2 STEP .1
80 CIRCLE 83,50,30,E,0
90 PAUSE 50
100 NEXT E
```

Fig. 9.6 Efecto de la *excentricidad* en la instrucción CIRCLE.

```
10 MODE 256:PAPER 0:INK 6:CLS
20 FOR A=0 TO PI STEP .1
40 CIRCLE 83,50,40,.5,A
50 PAUSE 20
60 NEXT A
```

Fig. 9.7 Variación de la inclinación de las elipses.

En la figura 9.7 se muestra el efecto de variar la inclinación del eje mayor. El ángulo se mide con respecto a la vertical y se expresa en radianes, no en grados. El radián es una unidad de medida más «natural» que el grado. La relación entre radián y grado es tal que PI (=3,1416) radianes equivalen a 180 grados; no obstante, estos detalles no son necesarios para apreciar el efecto de la instrucción CIRCLE. El QL entiende por PI el número π, como demuestra el programa. La línea 10 establece los colores de PAPER e INK; obsérvese que hemos incluido CLS para que PAPER 0 afecte a toda la pantalla. El bucle que empieza en la línea 20 varía el ángulo entre 0 y PI radianes (entre 0 y 180 grados) a saltos de 0,1 radianes. Para cada valor del ángulo se dibuja una elipse de excentricidad 0,5; en cada pasada del bucle se introduce una pausa para que el lector pueda observar la evolución del programa. Con una pequeña modificación se puede dar la ilusión de una elipse que gira

```
10 MODE 256:PAPER 0:CLS
20 FOR A=0 TO PI STEP .1
25 INK 6
40 CIRCLE 83,50,40,.5,A
52 INK 0
54 CIRCLE 83,50,40,.5,A
60 NEXT A
```

Fig. 9.8 Animación de gráficos mediante el cambio de tintas.

(Fig. 9.8). En este caso dibujamos las elipses alternadamente con INK 6 e INK 0, lo que equivale a borrar cada elipse una vez dibujada. El efecto óptico es el de una elipse en rotación.

Según el manual, se puede utilizar la instrucción CIRCLE_R para manejar coordenadas relativas (es decir, coordenadas que tienen por origen el punto en el que se encuentra el cursor), pero en mi QL provoca el mensaje «bad line». Sin embargo, otras instrucciones que manejan coordenadas relativas sí funcionan correctamente; las explicaremos en su momento.

ARCOS

El QL permite dibujar trozos de circunferencias, esto es, arcos. El nombre de la instrucción es obvio: ARC (o ARC_R para coordenadas relativas). ARC lleva como parámetros dos pares de coordenadas y un ángulo. Las dos primeras coordenadas son las del punto origen del arco; las dos siguientes son las del otro extremo; el último parámetro es el ángulo que el cursor tiene que describir al dibujar el arco. Si este ángulo es cero, el «arco» será un segmento de recta; si el ángulo es PI radianes, el arco es una semicircunferencia. Un ángulo expresado en grados se puede convertir en radianes mediante la función RAD. Así, 180 grados es RAD(180) radianes.

```
20 CLS
30 FOR N=5 TO 155 STEP 10
40 ARC N,30 TO N+10,30,RAD(180)
50 NEXT N
60 FOR N=5 TO 155 STEP 20
70 ARC N,50 TO N+20,50,RAD(300)
80 ARC N,80 TO N+20,80,RAD(60)
90 NEXT N
```

Fig. 9.9 Instrucción ARC. No todos los ángulos son válidos en esta instrucción.

La figura 9.9 da una idea de lo que se puede conseguir con la instrucción ARC. El primer bucle dibuja un conjunto de arcos que unen puntos distantes entre sí diez unidades; el ángulo es RAD(180), de forma que los arcos son semicircunferencias. En teoría, para dibujar arcos menos pronunciados bastaría con poner ángulos menores; pero la máquina es bastante inconsistente en esto. Para ángulos menores que 60 grados todo va bien; con ángulos comprendidos entre 60 y 180 grados se producen unos efectos muy raros, casi artísticos. Los ángulos mayores que 180 grados no dan problemas, como muestra la línea 70. Esta instrucción se puede abreviar si el origen del arco es la posición actual del cursor. Por ejemplo, la línea 40 se reduce a

ARC TO N+10,30,RAD(180)

Como hemos indicado, las coordenadas que se utilizan con ARC_R son relativas. Por ejemplo, si el cursor está en el punto 85, 50 y se especifica un punto de coordenadas relativas 10, 10, las coordenadas absolutas de este punto serán 95, 60 (o sea, 85 + 10, 50 + 10). Las coordenadas relativas pueden ser también números negativos. La figura 9.10 ilustra la utilización de ARC_R. La línea 20 se sirve de ARC para colocar el cursor en la posición

```
10 CLS
20 ARC 5,10 TO 15,10,RAD(60)
30 FOR N=1 TO 8
40 ARC_R 0,0 TO 10,10,RAD(60)
50 NEXT N
60 PAUSE 100
70 FOR N=1 TO 8
80 ARC_R 0,0 TO -10,-10,RAD(60)
90 NEXT N
```

Fig. 9.10 Ejemplo de utilización de ARC_R.

deseada (más adelante veremos otra forma de hacerlo). En el primer bucle se dibuja con ARC_R un conjunto de arcos enlazados. El segundo bucle cierra los arcos anteriores, para lo cual las coordenadas relativas son ahora las opuestas. Al igual que con la instrucción CIRCLE, con una sola instrucción ARC se pueden dibujar varios arcos, siempre que los grupos de parámetros estén separados por signos de punto y coma.

DIBUJO DE RECTAS

La instrucción LINE tiene diversas formas. En la primera de ellas lleva como parámetros un par de coordenadas; el efecto es mover el cursor hasta el punto especificado, sin dibujar nada. Por ejempo, LINE 83,50 sitúa el cursor en el centro de la pantalla (del televisor). Esta versión se puede utilizar para colocar el cursor donde convenga antes de una instrucción ARC.

```
10 CLS
20 LINE 83,90 TO 10,50
30 LINE TO 83,10 TO 160,50 TO 83,90
```

Fig. 9.11 Dibujo de rectas con LINE.

Pero, como el nombre sugiere, lo más frecuente es utilizar LINE para dibujar rectas. Por ejemplo, LINE 30,50 TO 40,20 traza una recta entre esos dos puntos, entendidos como coordenadas absolutas. Si se omite el primer par de coordenadas, la recta se traza desde la posición actual del cursor hasta el punto especificado (por ejemplo, LINE TO 83,50). La figura 9.11 da varios ejemplos de utilización de LINE; también ilustra cómo programar va-

```
10 CLS
20 LINE 83,50
30 FOR N=1 TO 50
40 INK RND(0,7)
50 LINE 83,50 TO RND(10,150),RND(10,90)
60 NEXT N
70 PAUSE 200:PAPER 0:CLS
80 LINE 83,50
90 FOR N=1 TO 50
100 INK RND(1,7)
110 A=RND(-0,1):IF A=0 THEN A=-1
120 LINE_R 0,0 TO A*RND(1,10),A*RND(1,10)
130 NEXT N
```

Fig. 9.12 Dibujo de rectas aleatorias con LINE y LINE_R.

rias rectas en una sola instrucción. Con LINE_R las coordenadas serán relativas a la posición del cursor. El programa de la figura 9.12 dibuja rectas desde el centro de la pantalla hasta posiciones aleatorias mediante LINE. Muy distinto es el efecto que se consigue con LINE_R. Cada recta queda enlazada con la anterior. La línea 110 genera el número aleatorio A, que puede ser +1 o −1; este número se utiliza en 120 para hacer aleatorio el sentido de la recta, cuya longitud también es aleatoria. El resultado es un «camino aleatorio», similar al que describiría una mosca chiflada.

GRÁFICOS DE TORTUGA

El dibujo con coordenadas tiene sus desventajas. Supongamos que queremos dibujar varios cuadrados. Para dibujarlos con instrucciones LINE tendremos que calcular las coordenadas de los vértices de todos ellos. Hay una forma de evitar ese trabajo: los gráficos de tortuga.

Definamos un cuadrado de la siguiente forma. Tomamos un punto; dibujamos una recta hacia arriba de longitud K puntos. Giramos 90 grados hacia la derecha y trazamos otra recta de igual longitud. Repetimos el proceso hasta completar el cuadrado. Si queremos un cuadrado pequeño, hacemos pequeño el valor de K; si queremos un cuadrado grande, damos a K un valor mayor. Esto tiene además otra ventaja: supongamos que queremos que el cuadrado esté inclinado, es decir, que sus lados no sean paralelos al borde de la pantalla. Para ello basta dar la inclinación deseada a la primera recta; el resto del procedimiento sigue siendo el mismo. Este sistema se llama a veces *coordenadas polares*; las coordenadas que hemos utilizado antes son

coordenadas cartesianas. Este nuevo método se denomina también *gráficos de tortuga*, porque las instrucciones son similares a las que se utilizan para controlar una «tortuga»: una caja dirigida por control remoto y movida por un motor que tiene una pluma con la que dibuja. Con las instrucciones adecuadas se puede hacer que la tortuga avance, retroceda, gire, suba o baje la pluma, dibuje, etc. Abordaremos este método de dibujo describiendo las instrucciones MOVE (mover), TURN (girar) y PEN (pluma).

MOVIMIENTOS DE LA TORTUGA

Empecemos por las instrucciones PEN. PENUP «levanta la pluma», es decir, impide que la tortuga dibuje al moverse. PENDOWN baja la pluma, lo que significa que en lo sucesivo la tortuga dejará rastro. TURN cambia la orientación de la tortuga, es decir, la dirección en la que ésta «mira». La orientación inicial es de izquierda a derecha, que es la que se toma como cero grados (la tortuga no maneja radianes). Si se pierde el control de la tortuga, se la puede orientar con TURNTO, por ejemplo, TURNTO 0. Las instrucciones TURN y TURNTO deben ir seguidas del ángulo de giro (en grados). Para girar en el sentido de las agujas del reloj el ángulo ha de ser negativo; los ángulos positivos especifican giro en sentido contrario al de las agujas del reloj. TURNTO impone un giro absoluto, lo que significa que los ángulos están referidos a la orientación inicial. TURN produce giros relativos, esto es, giros con respecto a la orientación actual de la tortuga.

Veamos un ejemplo sencillo: cómo trazar rectas verticales en la pantalla. De hecho, esto es más fácil con instrucciones LINE, pero vamos a hacerlo con la tortuga (Fig. 9.13). La línea 20 sitúa el cursor en 20, 10, que será por lo tanto el punto de partida de la tortuga. La línea 30 baja la pluma para

```
10 CLS
20 LINE 20,10
30 PENDOWN
40 TURNTO 90
50 MOVE 20
60 PENUP
70 TURNTO 0
80 MOVE 20
90 TURN -90
100 PENDOWN
110 MOVE 20
```

Fig. 9.13 Dibujo de dos rectas verticales con instrucciones de tortuga.

```
10 CLS
20 LINE 30,50
30 PENDOWN
40 FOR N=1 TO 12
50 MOVE 10
60 TURN 30
70 NEXT N
80 PAUSE 200
90 CLS
100 LINE 83,50
110 TURN RND(0,360)
120 FOR N=1 TO 100
130 MOVE RND(1,20)
140 TURN RND(1,270)
150 NEXT N
```

Fig. 9.14 Aplicación de los gráficos de tortuga a la creación de formas geométricas.

que en lo sucesivo la tortuga dibuje al moverse. La línea 40 orienta la tortuga hacia arriba (ángulo positivo porque el giro es antihorario). La línea 50 mueve la tortuga 20 unidades hacia arriba y traza la correspondiente recta. La línea 60 levanta la pluma; la línea 70 pone nuevamente la tortuga mirando hacia la derecha mediante el giro absoluto TURNTO 0 (también podríamos haber puesto TURN −90). La línea 80 mueve la tortuga 20 unidades hacia la derecha, sin dibujar. La línea 90 pone la tortuga mirando hacia abajo. La línea 100 baja la pluma para que la última instrucción dibuje otra recta.

Esto es tedioso porque la tortuga no es el mejor medio de realizar dibujos de este tipo. Su utilidad se manifiesta, en cambio, en el dibujo de figuras complejas, sobre todo formas geométricas. La figura 9.14 da un ejemplo de la velocidad y sencillez con que la tortuga traza gráficos complicados. Las líneas 30 a 70 dibujan un polígono de 12 lados; un bucle de doce pasadas hace girar 360/12 grados cada vez. En la segunda parte del programa se hace que la tortuga describa un camino aleatorio. Aunque la tortuga se salga de la pantalla no se produce mensaje de error. Se pueden definir procedimientos en los que se mezclen estas instrucciones para crear otras nuevas con las que simular la capacidad gráfica del lenguaje LOGO.

PLANIFICACIÓN DE LOS GRÁFICOS

Antes de continuar explorando las instrucciones gráficas del QL vamos a dar otro consejo sobre planificación del trabajo. Cuanto más complejos sean los gráficos que el lector diseñe, más cuidadosa debe ser su planificación. Es

conveniente dibujar sobre papel milimetrado, o bien sobre papel vegetal encima de papel milimetrado. Cualquiera que sea el tipo de gráficos que realice, siempre tendrá que utilizar coordenadas de puntos; si se dibuja en papel milimetrado es más fácil apreciar la forma y la escala del dibujo que si las coordenadas se ponen a ojo. Los gráficos improvisados pueden ser divertidos, pero a la larga constituyen una pérdida de tiempo y de paciencia.

REPRESENTACIÓN GRÁFICA

La representación gráfica es una de las aplicaciones más importantes de la capacidad gráfica de un ordenador si éste va a ser utilizado con programas comerciales o educacionales. Una *gráfica* (en femenino) es un conjunto de puntos que pueden ser unidos entre sí y que contienen alguna información. El QL dibuja puntos de gráficas mediante las instrucciones POINT (en coordenadas absolutas) y POINT_R (en coordenadas relativas). El color del dibujo es el color actual de INK.

```
10 PAPER 7:CLS
20 FOR X=1 TO 165
30 Y=50+50*SIN(2*PI*X/165)
40 INK 1:POINT X,Y
50 Y=50+50*(SIN(2*PI*X/165)^2)
60 INK 2:POINT X,Y
70 Y=50+50*(SIN(2*PI*X/165)^3)
80 INK 3:POINT X,Y
90 NEXT X
```

Fig. 9.15 Representación gráfica de tres curvas.

El programa de la figura 9.15 dibuja tres gráficas al mismo tiempo, y a velocidad considerable. En la línea 20 se establece el margen de X de forma que cubra toda la pantalla. Para cada valor de X se calculan tres valores de Y. El primero se obtiene del seno del ángulo cuyo valor es X/165 radianes; el segundo, del cuadrado de ese seno; y el tercero, del cubo. Los tres puntos se dibujan en las líneas 40, 60 y 80 con tres tintas diferentes. El número 50 que interviene en las fórmulas de Y tiene como objeto cubrir la pantalla con las gráficas. El seno de un ángulo tiene un valor que está comprendido entre -1 y $+1$. Pero un desplazamiento de una unidad en el eje Y no es muy impresionante, y por eso lo multiplicamos por 50. El margen de variación se convierte así en -200, $+200$. Ahora bien, en esta pantalla no podemos di-

bujar un punto cuya coordenada Y sea −50; sumando 50 a todos los valores de Y hacemos que el margen vaya de 0 a 100. Otro detalle que se debe observar es la expresión que hemos puesto para el ángulo: 2*PI*X/165. Esto es necesario para que el ángulo quede en radianes. Los ángulos de las funciones trigonométricas varían entre 0 y 2*PI radianes; cuando X es 0, 2*PI*X/165 es también cero, lo que da cero radianes en el extremo izquierdo de la gráfica. Cuando X es 165, 2*PI*X/165 es 2*PI radianes, precisamente lo que queríamos.

Después de ejecutar el programa haga la siguiente prueba: teclee SCALE 500,0,0, pulse ENTER y ejecute otra vez el programa. Las gráficas son mucho más estrechas y han quedado arrinconadas al lado izquierdo de la pantalla. La instrucción SCALE hace lo que su nombre indica: modifica la escala de la pantalla gráfica. Con SCALE 500,0,0 hacemos que el margen de Y sea de 0 a 500, y que el valor máximo de X se modifique en esa misma proporción (aproximadamente 5*165). La gráfica que obtenemos es por lo tanto muy pequeña; con SCALE 10,0,0 sería casi invisible. SCALE facilita la representación gráfica al permitirnos que nos olvidemos de los números de escala que tuvimos que introducir en la figura 9.15.

```
10 PAPER 7:CLS
20 SCALE 2,0,-1
30 FOR N=0 TO 3 STEP .1
40 POINT N,SIN(2*PI*N/3)
50 NEXT N
```

Fig. 9.16 Utilización de SCALE.

En la figura 9.16 hemos hecho SCALE 2,0, −1. El primer parámetro es el valor máximo de la coordenada Y; hemos puesto 2, ya que el seno varía entre −1 y +1. Los dos últimos parámetros son las coordenadas, en la nueva escala, del extremo inferior izquierdo de la pantalla. Hemos puesto 0, −1 para que cuando el seno sea −1 el punto correspondiente se represente en el borde inferior de la pantalla. No modificamos la coordenada X del origen de la pantalla, porque queremos representar ángulos de 0 a 2*PI, como antes. La introducción de SCALE simplifica las expresiones. Para volver a la situación normal hay que teclear SCALE 100,0,0.

La instrucción RECOL afecta tanto a los gráficos como al texto y sirve para cambiar todos los colores de la pantalla. RECOL va seguida de ocho parámetros; cada uno de ellos indica qué nuevo color va a sustituir al antiguo. El orden es: azul, rojo, magenta, verde, cyan, amarillo, blanco, negro. Por ejemplo, se pueden convertir todos los colores al negro mediante la ins-

trucción RECOL 0,0,0,0,0,0,0,0. Esto no es tan inútil como puede parecer: en efecto, después de ejecutar esta instrucción se puede dibujar un gráfico invisible y hacerlo aparecer súbitamente con RECOL 1,2,3,4,5,6,7,0. También se puede dibujar invisiblemente un conjunto de figuras y luego hacerlas aparecer una a una con la instrucción RECOL adecuada. Ésta es una forma sencilla de dar la ilusión de movimiento sin gran esfuerzo por parte del programador.

10

Los sonidos del QL

La capacidad de producir sonidos es una característica esencial de todos los ordenadores modernos. El sonido del QL proviene del altavoz que tiene incorporado, que es considerablemente mejor que el del Spectrum. Sin embargo, no parece que en el QL haya control de volumen. Además, el margen de notas disponibles es bastante limitado en comparación con el de otras máquinas.

El fenómeno que denominamos *sonido* es el resultado de las variaciones rápidas de presión en el aire cercano a nuestros oídos. El oído no las detecta si no son suficientemente rápidas. La frecuencia de las variaciones se mide en *hertzios* (ciclos por segundo). Un ciclo de una onda es el tiempo que ésta tarda en aumentar la presión hasta el máximo, volver a la presión normal,

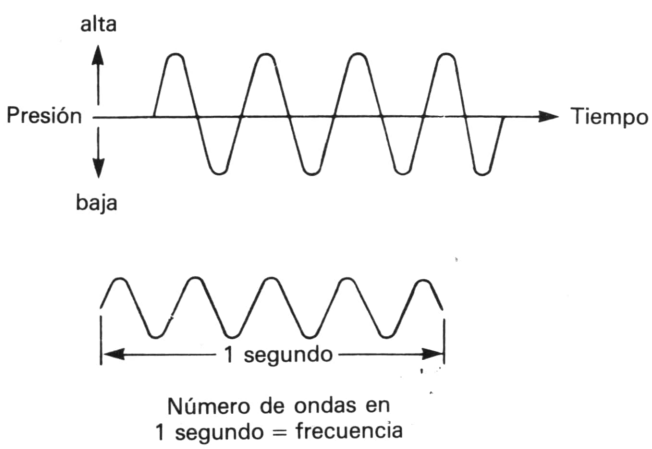

Fig. 10.1 Representación gráfica de las ondas de sonido. La presión varía con el tiempo; el número de cambios por segundo es la frecuencia.

disminuir la presión hasta el mínimo y volver por segunda vez a la presión normal (Fig. 10.1). La razón por la que utilizamos el término *onda* es la forma que tiene la gráfica de la presión en función del tiempo, similar a la de las ondas en la superficie del agua.

La *frecuencia* de un sonido es el número de hertzios (Hz), es decir, el número de ciclos de variación de la presión por segundo. Si la frecuencia es inferior a 20 Hz, el sonido es inaudible, aunque puede tener otros efectos. El margen de frecuencias audibles se extiende desde los 20 hasta más de 15000 Hz. La frecuencia de las ondas es lo que percibimos como «tono» de una nota. Las frecuencias de 80 a 120 Hz corresponden a lo que entendemos por tonos graves; los tonos agudos son los que tienen frecuencias superiores a 400 Hz.

La amplitud de las variaciones de presión determina la *intensidad* de una nota. La *amplitud* de una onda es la máxima variación de la presión con respecto al valor normal. Para controlar plenamente la generación del sonido tenemos que especificar la amplitud, la frecuencia y la forma de la onda, así como la forma de variación de la amplitud durante el tiempo en que la nota esté sonando.

Al escribir música el compositor tiene que especificar la intensidad, la duración y el tono de cada nota. En la música escrita la intensidad se indica con las letras *f* (forte, intenso) y *p* (piano). Por ejemplo, *fff* significa fortísimo (muy intenso) y *ppp* significa pianísimo (muy suave). La duración de las notas se indica de dos formas distintas. Una de ellas se basa en el funcionamiento del metrónomo. Este aparato es un generador de sonido que produce un «tic» a intervalos regulares; del ajuste del metrónomo depende el número de batidos que realiza por minuto. La unidad de duración de las notas es la nota *negra*, de forma que el ajuste del metrónomo decide cuántas deben sonar por minuto. La duración de una nota se expresa por comparación con esta unidad. Así, una *mínima* o *blanca* dura el doble que una negra, y una *semibreve* o *redonda* dura el doble que una mínima. Una *corchea* es la mitad

Símbolo	Duración	Nombre
𝅘𝅥𝅯	1/8	Fusa
𝅘𝅥𝅰	1/4	Semicorchea
𝅘𝅥𝅮	1/2	Corchea
𝅘𝅥	1	Negra
𝅗𝅥	2	Blanca
𝅝	4	Redonda

Fig. 10.2 Símbolos que se utilizan en las partituras para indicar la duración de las notas.

de una negra; una *semicorchea*, la cuarta parte. En la figura 10.2 se muestran las posibles duraciones de las notas y los signos correspondientes. Hay otros signos que se emplean para representar la duración de los silencios (Fig. 10.3). El otro método de especificar la duración de las notas es bastante menos preciso; utiliza palabras italianas tales como lento, moderato y allegro.

Símbolo del silencio	Duración
𝄿	1/4
𝄾	1/2
𝄽	1
𝄼	2
𝄻	4

Fig. 10.3 Símbolos de los silencios en las partituras.

El tono de una nota se indica por la altura a la que se la escribe en el *pentagrama*. La música para piano se escribe en dos pentagramas paralelos, cada uno de los cuales está formado por cinco rayas y cuatro espacios. El pentagrama superior es el de agudos (clave de sol) y el inferior, el de graves (clave de fa). La música para instrumentos que no tienen teclado se escribe en un

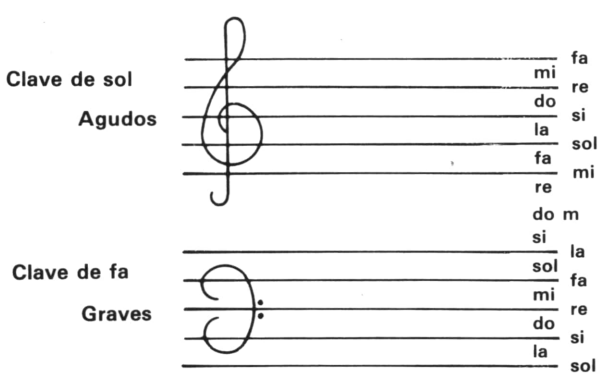

Fig. 10.4 Los dos pentagramas, con los nombres de las notas.

solo pentagrama. Para el piano y el órgano se utilizan, como hemos dicho, dos pentagramas. Entre ellos queda el espacio para la nota *do media*. En el piano esta nota se obtiene de la tecla que está en el centro del teclado. En la figura 10.4 se muestran los dos pentagramas.

Las notas que aparecen en la figura 10.4 se disponen en grupos de 8. Cada grupo forma una *octava*. También hay notas de tono intermedio entre cada dos de estas notas: son los *semitonos*. En la música occidental la octava se

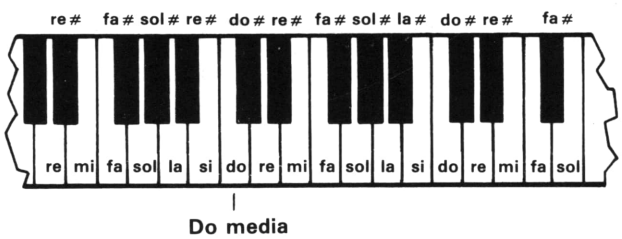

Fig. 10.5 Parte del teclado del piano, en la que se indica la posición de la nota do media. Sólo hay un semitono entre sí y do y entre mi y fa.

divide en un total de doce notas, entre tonos y semitonos. En la figura 10.5 se indica su colocación en el teclado del piano. Los semitonos son los correspondientes a las teclas negras. En las partituras los semitonos se indican mediante los signos # (sostenido) y ♭ (bemol). Un sostenido indica un semitono por encima del tono de la nota; un bemol, un semitono por debajo. En el piano, un semitono por encima de una nota es igual a un bemol por debajo de la siguiente; así, do sostenido es lo mismo que re bemol. Esto no es necesariamente cierto en otros instrumentos.

INSTRUCCIONES DE SONIDO DEL QL

El QL dispone de una instrucción de sonido que se puede utilizar de distintas formas, desde la más sencilla hasta la más compleja. Todo depende del efecto que se desee conseguir; no se requiere gran planificación si lo que se pretende es producir una nota de aviso. A medida que se va adquiriendo experiencia, se le van encontrando aplicaciones más interesantes. Empecemos con la simple generación de notas.

La instrucción BEEP va seguida de dos números. El primero es el *número de duración*, que determina durante cuánto tiempo sonará la nota. El margen de este parámetro va de 1 (demasiado corto para ser audible) a 32767,

una nota muy larga. El valor 0 tiene un significado especial: hace que la nota suene hasta que se la anule. Estos parámetros pueden ser positivos o negativos, pero el ordenador sólo tiene en cuenta su valor absoluto.

El segundo parámetro de BEEP es el *número de tono*. Este número controla el tono de la nota; su margen va de 0 (nota más aguda) a 255 (nota más grave). En la figura 10.6 se da la equivalencia aproximada entre los números de tono y las notas y escalas del piano. Este instrumento es el más conocido y en su teclado se puede tocar muy fácilmente una sucesión de ocho notas

Número	Nota	Número	Nota
11	Si (octava por encima de do media)	35	Si (debajo de do media)
		38	La #
12-13	Si	41	La
14	La #	44	Sol #
15	La	47	Sol
17	Sol #	51	Fa #
18	Sol	55	Fa
20	Fa #	59	Mi
22	Fa	63	Re #
24	Mi	67	Re
26	Re #	72	Do #
28	Re	77	Do, (octava por debajo de do medio
31	Do #		
33	Do media		

Fig. 10.6 Números de tono en el QL. Esta lista corresponde a las dos octavas adyacentes a la nota do media.

concretas, la llamada «escala de do mayor». Esta escala empieza en do media y termina en otra nota do, pero una octava más arriba. La instrucción BEEP reproduce bastante bien las notas que están por debajo de do media, pero no las superiores; esto limita la utilidad de esta instrucción en la programación de música.

Comencemos la investigación de BEEP con unas notas sencillas. En el programa de la figura 10.7 se generan tres notas separadas por una pausa. Estas pausas son, como veremos, muy importantes. Al ejecutar el programa se oirá una do media corta, luego la do de la siguiente octava, más larga, y finalmente la do de la octava inferior a do media.

Las pausas son necesarias si se quiere que las notas suenen por separado.

```
10 BEEP 2000,33
15 PAUSE 40
20 BEEP 20000,11
25 PAUSE 70
30 BEEP 1000,77
```

Fig. 10.7 Generación de notas aisladas con BEEP.

Si se elimina la línea 25, se oye la primera nota sola, pero las otras dos suenan casi simultáneamente, y sólo durante el tiempo correspondiente a la duración de la más corta. Si ahora se elimina la línea 15, las tres notas suenan durante sólo un instante. Lo que ocurre es que la primera nota suena hasta que el ordenador ejecuta la instrucción de la segunda; ésta cancela la primera y a su vez es suprimida por la tercera. En todo momento se puede cancelar una nota que esté sonando sin más que ejecutar otra instrucción BEEP, incluso sin parámetros.

El siguiente paso va a consistir en generar una escala musical. En el manual del QL no se da la correlación entre los números de tono y las notas, por lo que el lector tendrá que basarse en la figura 10.6. El programa de la figura 10.8 ejecuta la escala de do mayor, empezando por la nota do media

```
10 FOR N=1 TO 8
20 READ TONO
30 BEEP 20000,TONO
40 PAUSE 60
50 NEXT N
60 DATA 33,28,24,22,18,15,12,11
```

Fig. 10.8 La escala de do mayor, interpretada por el QL. Algunas notas no se pueden obtener con precisión.

y terminado en la do de la octava siguiente. El salto de una nota a la misma nota de la octava siguiente representa duplicar la frecuencia; dividir por dos la frecuencia equivale a descender una octava. Los números de tono del QL funcionan inversamente; es decir, para subir la frecuencia hay que bajar el número de tono. Como se puede observar, el número de tono de do media es 33, mientras que el de la nota do de la octava siguiente es 11. No hay una relación sencilla entre los números de tono y las frecuencias de las notas generadas por BEEP.

A diferencia de otros ordenadores modernos, el QL sólo tiene un canal de sonido, y por consiguiente no permite producir armonía. Por lo tanto no tiene demasiado interés seguir hablando de la programación de la música, de

modo que nos limitaremos a indicar cómo se pueden generar melodías sencillas. Si el lector no es compositor de música, lo mejor que puede hacer es tomar como guía una partitura. La música más adecuada es la escrita para violonchelo o para voz de barítono. Si utiliza una partitura para piano, tendrá que averiguar qué parte contiene la melodía, y para ello hay que saber leer música. Muchas composiciones musicales requieren notas agudas que no se pueden programar en el QL. Evite la música escrita para instrumentos tales como el clarinete o el bajón, en las que las notas suenan de forma distinta de como lo harían en otro instrumento que ejecutara la misma música.

La mejor técnica de programación consiste en construir un bucle que lea en líneas DATA los parámetros de BEEP y la duración de las pausas. El parámetro de PAUSE debe contener una variable, con lo que se podrá acelerar o decelerar la música sin tener que cambiar los datos. Cuando se empieza a practicar con música es conveniente poner los datos de cada nota en una línea DATA distinta. La música de piano suena mejor si se introduce cierta separación entre notas; pero la de órgano no necesita tal cosa. La experiencia le permitirá refinar los programas una vez aprendidas las técnicas básicas.

Veamos por ejemplo el programa de la figura 10.9, que ejecuta una melodía sencilla. Este programa se escribió siguiendo una partitura. Lo funda-

```
5 RESTORE
10 S=7:R=2000
20 FOR J=1 TO 5
30 READ N,D,P
40 BEEP D*R,N
50 PAUSE P*S
60 NEXT J
100 DATA 33,4,4
110 DATA 33,2,2
120 DATA 33,2,2
130 DATA 11,6,6
140 DATA 12,2,2
150 DATA 12,2,2
160 DATA 15,6,9
170 DATA 15,2,2
180 DATA 18,4,4
190 DATA 24,4,4
200 DATA 24,4,3
210 DATA 28,2,2
220 DATA 33,2,2
230 DATA 24,8,8
240 DATA 28,4,4
```

Fig. 10.9 Generación de una tonada sencilla.

mental del programa es un bucle que lee las variables N, D y P, que representan el tono, la duración y la pausa de cada nota. Los parámetros de BEEP son proporcionales a estas variables. Los valores de las líneas DATA son los que reproducen con mayor fidelidad la melodía. Experimentando con las constantes de la línea 10 se controla el tempo de la música.

EFECTOS ESPECIALES

La instrucción BEEP puede producir, aun en esta forma tan sencilla, gran variedad de efectos sonoros. Empecemos con una serie de notas de frecuencia creciente. Este efecto puede servir como llamada de atención o aviso de que «algo está a punto de suceder». Consideremos el ejemplo de la figura 10.10. El bucle recorre el margen de 255 a 0, todos los valores posibles del

```
10 FOR N=255 TO 0 STEP -1
20 BEEP 2000,N
30 NEXT N
```

Fig. 10.10 Programación de una nota de tono creciente.

número de tono. El número de duración es fijo, 2000. Esta duración puede parecer excesiva, pero aunque se ponga un número más pequeño no mejora apreciablemente la velocidad de ejecución del bucle. Los números de duración más cortos producen notas menos agradables. Para acortar el tiempo de ejecución de la serie de notas se puede hacer que N varíe entre 100 y 0.

El programa de la figura 10.11 produce un trino. Un sonido de este tipo es útil para llamar la atención del usurio o para anunciar la inminencia de un suceso en un juego. Por alguna razón, las notas de este tipo atraen mejor la atención que las notas sencillas; por eso se las utiliza en los teléfonos modernos. El programa consiste en un bucle que hace sonar 100 pares de notas;

```
10 FOR N=1 TO 100
20 BEEP 2000,33
30 PAUSE 2
40 BEEP 2000,34
50 PAUSE 2
60 NEXT N
```

Fig. 10.11 Programación de un trino.

las notas son cortas y están separadas por las pausas de las líneas 30 y 50. Los números de tono son 33 y 34. Cuanto más agudo sea el tono, más eficaz será el sonido en cuanto a llamar la atención. Pruebe, por ejemplo, con números de tono 3 y 4. En este ejemplo el número de duración es 2000, el mínimo aceptable. El número 1000 da un resultado bastante peor, especialmente con números de tono 3 y 4.

OTRAS FORMAS DE LA INSTRUCCIÓN BEEP

La instrucción BEEP del QL produce efectos sonoros mucho más variados que la del Spectrum. Estas nuevas funciones se consiguen poniendo otros parámetros después de los números de duración y de tono. Estos parámetros definen la *envolvente* de la nota. Su efecto es difícil de describir con palabras; así pues, ejecute los programas y escuche con atención.

Antes de nada, expliquemos qué es una envolente. Los sonidos que produce la instrucción BEEP, tal como la conocemos, tienen amplitud y frecuencia constantes; es decir, su intensidad y su tono no varían durante la ejecución de la nota. Pero las notas producidas por los instrumentos musicales no son así. Por ejemplo, una nota de piano es más intensa al principio, inmediatamente después de que el martillo golpee la cuerda, y luego se atenúa muy deprisa. Esto es lo que hace tan inconfundibles los sonidos del piano. Otros instrumentos producen las notas de forma muy diferente; todos ellos producen notas que no son tonos puros, sino mezclas de diversas frecuencias. Estas diferencias son las que permiten distinguir un piano de un violín, y éste de una flauta, aunque los tres estén ejecutando la misma nota. Una gráfica de la amplitud de una nota en función del tiempo es lo que se denomina *envolvente de volumen* de la nota. El QL no permite controlar este tipo de envolvente.

No obstante, se pueden conseguir efectos muy interesantes controlando la forma de varición del tono de las notas. Este control se realiza con los parámetros tercero al sexto de la instrucción BEEP. El tercer parámetro es el segundo número de tono, es decir, el valor que se debe alcanzar al variar el tono. El cuarto parámetro especifica la velocidad de variación del tono; es el número GRAD_X. En ciertas notas necesitamos que el tono vuelva al valor original; la velocidad de esta segunda variación se especifica en un nuevo parámetro, GRAD_Y. Finalmente, el último parámetro especifica cuántas veces se debe producir el paso del primer tono al segundo y de éste al primero; es el número «wrap». En la figura 10.12 se da un resumen del significado de estos parámetros y se especifican sus márgenes. Nótese que el margen de GRAD_X es de 0 a 15, mientras que el de GRAD_Y es de −8 a +7.

Segundo número de tono: el tono varía entre las frecuencias correspondientes a los dos números de tono. Margen: 0 a 255.

Grad X: especifica la velocidad de variación del primer tono al segundo. Margen: 0 a 15.

Grad Y: especifica la velocidad de variación del segundo tono al primero. Margen: −8 a +7 (el 0 no tiene efecto).

Wrap: especifica el número de cambios de un tono a otro. Margen: 0 a 15.

Ruido: añade ruido al sonido. Margen: 0 a 15.

Fig. 10.12 Resumen de los parámetros adicionales de la instrucción BEEP.

El programa de la figura 10.13 da una idea del efecto de los números GRAD_X y GRAD_Y. Los valores grandes de estos números hacen que el segundo tono predomine sobre el primero. Si ambos números son cero, lo que suena es una nota continua. Para utilizar los otros dos parámetros sin especificar ninguna velocidad de trino, en el lugar de los dos números GRAD se puede poner un 0.

```
10 FOR N=0 TO 15
20 PRINT"GRAD_X ES ";N;\"GRAD_T ES ";N-8
30 BEEP 32767,33,37,N,N-8,10
40 PAUSE 200
50 NEXT N
```

Fig. 10.13 Efecto de los parámetros GRAD.

El siguiente parámetro posible de BEEP es el «numero de ruido». Su efecto es, obviamente, añadir ruido al sonido. *Ruido* es una mezcla aleatoria de frecuencias, si bien en la práctica se observa que hay una frecuencia (o margen de frecuencias) que predomina sobre las demás. Escuche el sonido producido por el programa de la figura 10.14. En este caso el número de duración es 0, de modo que la duración será controlada por PAUSE. La variable del contador controla el número de ruido. Los valores más bajos no producen efecto apreciable; a partir de 8 el efecto es muy llamativo. Este efecto se puede combinar con las variaciones de tono descritas antes. Se puede añadir un último parámetro a la instrucción BEEP: el número «rand», cuyo margen va de −32767 a +32768, pero yo no he sido capaz de apreciar su efecto.

```
10 CLS
30 FOR N=0 TO 15
35 PRINT N
40 BEEP 0,33,44,0,0,14,N
50 PAUSE 100
60 NEXT N
70 BEEP
```

Fig. 10.14 Efecto del número de ruido.

No es fácil crear sonidos convincentes con la instrucción BEEP. Los ruidos de disparos, tambores, martillos neumáticos, etc., son difíciles de programar con BEEP. No voy a decir que sean imposibles, pero es evidente que requieren un tiempo de investigación que yo no he podido dedicarles al escribir este libro. En cambio, para sonidos más sencillos, la instrucción BEEP es adecuada.

Apéndice A

Edición

En este contexto, *editar* significa modificar algo que ha aparecido en la pantalla. Todo el contenido de una línea, *incluso su número*, se puede modificar por edición. El proceso de edición se puede realizar:

a) cuando se está tecleando una línea, antes de pulsar ENTER;
b) después de pulsar ENTER, pero antes de ejecutar el programa;
c) cuando el ordenador detecte un error durante la ejecución de un programa.

Analicemos estas tres posibilidades:

a) Durante la introducción de una línea se pueden utilizar todos los recursos de edición de línea. La edición se termina pulsando ENTER. Si se produce un mensaje de error al pulsar ENTER, la línea no ha quedado introducida en la memoria; en tal caso hay que teclear la línea completa.
b) Una vez introducida la línea, para editarla hay que prepararla tecleando la orden EDIT seguida del número de línea.
c) Cuando el programa se detiene con un mensaje de error, éste indicará el número de línea y el tipo de error. Esto no implica que la corrección haya que hacerla en la línea indicada. Por ejemplo, una línea DATA puede contener un dato incorrecto, pero el error se producirá en la línea que contenga la instrucción READ que lo lea.

ÓRDENES DE EDICIÓN

1. Teclas del cursor. Las dos teclas que están a la izquierda de la barra espaciadora son las teclas correctoras principales. Cuando se prepara la edición de una línea con la orden EDIT, el cursor queda situado al final de la

línea. Para hacerlo retroceder se utiliza la tecla de «cursor a la izquierda». La acción de las teclas de movimiento del cursor es repetitiva y muy rápida. Las teclas de movimiento vertical del cursor no tienen efecto en este caso.

2. Para insertar una o varias letras el cursor se debe colocar sobre la letra siguiente.

3. Para borrar una letra se sitúa el cursor sobre la letra siguiente y luego se pulsan al mismo tiempo CTRL y la tecla «cursor a la izquierda». También se puede poner el cursor sobre la letra anterior y pulsar CTRL y la tecla «cursor a la derecha».

4. El manual describe una serie de acciones para las que se combinan las teclas SHIFT y ALT con las de movimiento del cursor. Ninguna de ellas funciona en mi QL.

5. Al pulsar ENTER se da por concluida la edición, a no ser que se produzca un mensaje de error, pues en tal caso la línea no habrá sido modificada y habrá que corregirla nuevamente.

OTRAS INSTRUCCIONES

1. A veces es necesario borrar un grupo de instrucciones. Esta acción se realiza con DLINE. Por ejemplo, DLINE 10 TO 100 borra todas las líneas comprendidas entre la 10 y la 100, ambas inclusive.

2. Con la orden AUTO se puede hacer que el ordenador escriba automáticamente los números de línea; esto tiene utilidad en particular cuando se está copiando un programa en el que los números de línea están regularmente espaciados. Escribiendo AUTO y pulsando ENTER aparece el número 100 en la pantalla; cada vez que se pulse ENTER este número se irá incrementando de 10 en 10. Si por ejemplo se desea empezar en la línea 10 y que los incrementos sean de cinco en cinco, la orden necesaria es AUTO 10,5. Para cancelar esta orden se pulsa CTRL al mismo tiempo que la barra espaciadora.

Apéndice B

Conexión de una impresora

El QL sólo puede ser conectado a impresoras dotadas de interfaz «en serie». En este sistema de transmisión las señales se envían a la impresora agrupadas, en sucesiones de diez u once señales. Sin embargo, la mayor parte de las impresoras utilizan lo que se denomina «interfaz paralelo»; en este sistema las señales se transmiten simultáneamente, una por cada hilo. Normalmente a las impresoras «paralelo» se les puede acoplar un interfaz «serie», aunque puede resultar más económico acoplar al QL un interfaz que convierta su salida «serie» en «paralelo».

Si el lector va a conectar una impresora «serie», compruebe antes lo siguiente:

1. Velocidad de transmisión. Si la impresora puede funcionar a 4800 baudios, ajústela para esa velocidad.

2. Número de bits: póngala a 8.

3. Paridad (método de detección de errores): sin paridad.

4. Las conexiones entre el QL y la impresora deben permitir que ésta informe al ordenador cuándo está dispueta para recibir más datos.

5. En algunas impresoras se puede seleccionar la velocidad de transmisión desde el ordenador. Para prepararlas bastará con teclear la orden

BAUD 4800:OPEN # 3,SER1

6. Si la impresora no es ajustable de esta forma, después de SER1 habrá que incluir algunos otros códigos. Véase el manual del QL.

7. Una vez abierto el canal de la impresora, por ejemplo el # 3, se le puede enviar texto con las instrucciones PRINT # 3, LIST # 3, etc. Tenga en cuenta que la orden NEW cierra todos los canales. Esto es muy molesto cuando se quiere listar varios programas; en lugar de borrar los programas con NEW, se puede utilizar DLINE.

Índice

A

ABS 63
Actualización de ficheros 123
Adaptador en Y 3
ADATE 82, 83
Ángulo 146
Animación de gráficos 146
Amplitud 156
Apertura de ficheros 108
Apóstrofe 26
ARC 147
  margen de los ángulos 147
Archivo en microdrive 107
ASCII, código 67
Asignación 33
  de nombres a una lista 117
Asterisco 23
AT 27
AUTO 168

B

«Bad name», mensaje 11
Barra
  de dividir 22
  inclinada hacia la izquierda 30
Barra espaciadora 10
Bases de datos 91
BASIC 19
BEEP 158
Bemol 158
BLOCK 140
BOOT 17
BORDER 130, 137
Borrado
  de letras 168
  de líneas 168
Bucle(s)
  anidados 56
  externo 56
  interno 56
Byte 104

C

Cable de antena 3
Cadena
  literal 23, 67
  vacía 65
Cálculos numéricos 42
Camino aleatorio 149
Campo 106
Canales 40, 103
CAPS LOCK, tecla 10
Carácter 67
Carga y ejecución inmediata 17
Carga de programas 12
Cartuchos 6
  conservación 9, 16
Centrado de textos 28, 70
CHR$ 75
CHR$(0) 116
Cinta magnética 12
CIRCLE 143
Clavija de red 1
CLEAR 50, 125
CLOSE 110
CLS 17
CODE 75

Codificación de mensajes 76
Código de máquina 91
Color, instrucciones de 137
Coma 26
Comillas 23
Comparación
  de cadenas 77
  de números 60
Concatenación 37
Conexión(es)
  en la clavija de red 2
  ventana-canal 131
Conversión de variables 87
Coordenadas 128
  cartesianas 149
  esquema 129
  en gráficos 142
    esquema 142
    origen 143
  polares 149
  relativas 146
Copias de seguridad 12
COPY 111
Creación de ficheros 108
CSIZE 30
CTRL, tecla 10
Cursor 9, 134
  gráfico 148

D

DATA 41
DATE 82
DATE$ 82
Datos 104
DAY$ 82
Decrementar 43
Definición de funciones 47
Desplazamiento de la pantalla 132
DIM 79
Dimensionado de listas 79
  literales 80
DIR 14
Disco magnético 13
Disección de cadenas 70
Diseño de programas 91
Dispositivo fuente 112
  destino 112
Dispositivos 103
DIV 47
DLINE 168
Do media, nota 158
Dos puntos, signo de 25

E

EASEL, programa 12
Edición 167
Ejes de una elipse 145
Elipse 144
ELSE 52
Enchufe múltiple 4
END FOR 54
«End of file», mensaje 76, 116
ENTER, tecla 11
Entero positivo 21
Envolvente 163
ESC, tecla 10
Escala musical 160
Espacios, importancia de los 20
Excentricidad de las elipses 145
EXIT 51
Expresión 34

F

F1, tecla 7
Fichero(s) 105
  abiertos 111
  de acceso aleatorio 108
  secuenciales 108
Filas y columas 25
FILL 143
FILL$ 37
Fin de fichero, señal 111
Finalización 59
FLASH 134
FOR 54
Formateado de cartuchos 14

Fracciones 45
Frecuencia 155
Fuente de alimentación 1
Funciones
  literales 67
  numéricas 44
Fusible 1

G

GOTO 49, 62
Gráficos 141
  de tortuga 149

I

IF 51
Impresora 169
Incrementar 43
Inicializado de cartuchos 14
INK 138
INKEY$ 64, 118
INPUT 38
Inserción de letras 73, 168
INSTR 83
Instrucciones de repetición 49
Interconexión de ordenadores 103
Interfaz
  paralelo 169
  serie 169
Interrupción de un bucle 50
Interruptor 4

J

Juego de adivinación de números 63

K

KEYROW, números 65

L

Lectura de ficheros 114
Lectura de un fichero de
  nombres 117
LEN 69
Lenguajes de programación 19
LET 34
LINE 148
Línea multi-instrucción 26
LIST 17
Listas 78
LN 44
LOGO 151
LOG10 44
Longitud de cadenas literales 69

M

Manual 1
Matrices 80
  literales 81
Mayúsculas 10
MDV 9
Melodía sencilla 161
Memoria 12
Mensaje de copyright 7
Menú 85
Mezcla de tipos de variables 40
Microdive 6, 12, 13, 103
  piloto del 9
Minúsculas 10
MOD 47
MODE 141
Modificación de ficheros 123
Modificadores de escritura 134
Modo
  directo 19
  de programa 19
Monitor 2
MOVE 150
Mueble para el QL 5

N

Negra, nota 156
NEXT 54

Nombre
de fichero 16, 111
de variable 33
Nota de tono creciente 162
Notación científica 46
Numeración automática de líneas 168
Número(s)
de color 138
de columna 27
de duración en BEEP 158
de GRAD 163
de línea 15, 27
de ruido 164
de tono en BEEP 159
de «wrap» 163

O

Octava 158
OPEN 105
OPEN_IN 108
OPEN_NEW 108
Operaciones prohibidas 36
OR 52
Orden alfabético 77
Órdenes de edición 19
Orla para títulos 37
OVER 135
«Overflow», mensaje 46

P

Palabras reservadas 11
PAN 134
Pantalla 6, 8, 103
de textos, esquema 29
Parámetro 89
Paridad 169
PAPER 138
Pasada por el bucle 50
PAUSE 28
PENDOWN 150
Pentagrama 157
PENUP 150
«Pixels» 141
Planificación
de gráficos 151
de programas 92
POINT 152
Precisión 23
en los cálculos 45
en el dimensionado de listas 122
PRINT 20
Procedimientos 88
utilización de los 94
Programa(s) 19
estructurados 92
Punto y coma 23

R

Radián 146
READ 41
RECOL 153
Redondeo 45
Registro 106
REM, instrucción 14
REPEAT 49
Representación gráfica 152
RESET, botón 8, 12
Resolución 141
RESTORE 42, 76
Retorno del carro 11
RND 52
ROM, zócalo para 6
Ruido de la fuente de alimentación 2

S

Salida del bucle 51
SCALE 140, 153
SCR 131
SCROLL 132
Scrolling 24
SDATE 82
SELECT 86
Semitonos 158
Señal de fin de fichero 116

Separador inteligente 35
SHIFT, teclas 10

S

Signo
de dólar 67
de subrayar 108
de exclamación 35
Sintonización del televisor 6-9
Sonido 155
Sostenido 158
SQRT 44
STEP 57
STRIP 136
Subrayar, signo de 13
Subrutinas 88

T

Tampón 109, 111
Tanteador 98
Teclado 65
del piano 158
Teclas
del cursor 11
programables 10
Televisor 2
Tempo 162
THEN 51
TO 71
Toma de corriente 4
Tono 156
Totalizador 114
Trampa para errores 62
Trino 162
TURN 150
TURNTO 150

U

UNDER 137

V

Variable
entera 46
indexada 78
literal 35
local 48
Velocidad de transmisión 169
Ventana 127
WINDOW 128

=, signo 43

#, signo 105

&, signo 37

Avda. del Mediterráneo, 9. 28007 MADRID